TOWARDS AN EXPLANATORY DIVISION OF COMPETENCE AND PERFORMANCE: A LANGUAGE-INDEPENDENT PARSING SCHEME

By

Carl G. Alphonce
B.Sc., McGill University, 1988
B.A., McGill University, 1990

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
DEPARTMENT OF COMPUTER SCIENCE

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA
October 1992

Department of Computer Science
The University of British Columbia
2075 Wesbrook Place
Vancouver, Canada
V6T 1Z1

Date: October 13, 1992

Abstract

This dissertation defends in some small measure the thesis that there is a *universal* parsing model for natural languages. Such a model will apply, without change, cross-linguistically.

The defense of this thesis proceeds by finding solutions to some apparently insurmountable problems which arise from the interaction of a set of basic and seemingly uncontroversial assumptions concerning both the linguistic framework and the computational one. Some of the difficulties associated with parsing overtly and covertly derived unbounded $\overline{\mathrm{A}}$ dependencies, as instantiated in English and Chinese in particular, are explored; solutions which are psycholinguistically plausible are presented.

In further defense, it is claimed that there are certain linguistic phenomena, such as Relativized Minimality and some curious directional asymmetries in the movement rule move-α, which are better analyzed as artificats of *performance* constraints, rather than *competence* constraints; by removing some of the burden from the competence theory (the syntactic theory, in this case) and placing it on the performance theory (the parsing model), a more perspicuous model of human language-processing emerges. By delimiting more strictly the division between competence and performance in this manner, the adoption of a universal parsing scheme is made possible. This cross-linguistically applicable model makes strong computational and linguistic predictions, which are also explored.

Table of Contents

Abstract ii

List of Tables vii

List of Figures viii

Acknowledgements ix

1 Introduction 1

- 1.1 Thesis 1
- 1.2 Motivation 3
- 1.3 Outline 4

2 Linguistic Framework 5

- 2.1 A Modular Theory 5
- 2.2 A Multistratal Theory 6
- 2.3 $\overline{\text{X}}$ Theory 8
- 2.4 Basic Definitions 11
  - 2.4.1 Graphs and Trees 11
  - 2.4.2 Structural Relations 12
- 2.5 Affect-α and Chains 14
  - 2.5.1 Substitution 15
  - 2.5.2 Adjunction 16
  - 2.5.3 Chains 16

2.6 A-movement 18

2.6.1 θ Theory 18

2.6.2 Case Theory 19

2.6.3 Further Examples of A-movement 22

2.7 Extended Projection Principle 24

2.8 *pro* and *PRO* 25

2.9 Predication 26

2.10 $\overline{\text{A}}$-movement 28

2.10.1 *Wh*-movement 28

2.10.2 Operator Scope 30

2.10.3 Selectional Restrictions 31

2.10.4 Relative Clauses 33

2.11 Government 35

2.12 Relativized Minimality 36

2.13 The Empty Category Principle 40

2.14 Minimal Linguistic Assumptions 42

3 Computational Framework **43**

3.1 A General Model 43

3.1.1 The Marcus Parser 44

3.1.2 Licensing Parsers 46

3.1.3 Chunk Parsing 47

3.1.4 Other Details 49

3.1.5 Summary 51

3.2 Minimal Computational Assumptions 52

3.3 Psycholinguistic Evidence 53

3.3.1 Filler-Driven Parsing 53
3.3.2 General Parsing Principles 53

4 Problems and Solutions **55**

4.1 Relativized Minimality Effects 56
4.2 Brief Overview of the Parser's Operation 58
4.3 A-type Movements 63
4.4 *Wh*-movement 70
4.4.1 S-Structure *Wh*-movement 70
4.4.2 LF *Wh*-movement 78
4.5 Revised Parsing Model 91
4.6 Empty Operator Constructions 92

5 Implications **94**

6 Related Work **97**

6.1 Three Approaches 97
6.1.1 Language-Specific Parsing 98
6.1.2 Parameterized Parsing 98
6.1.3 Common Parsing Scheme 100
6.2 Other Research 101
6.2.1 Long-Distance $\overline{\text{A}}$-dependencies at S-Structure 101
6.2.2 A Grammar Tester 101

7 Future Work **102**

8 Conclusion **104**

Bibliography **106**

A The Parser **110**

A.1 Implementation 110

A.2 Examples 111

List of Tables

2.1 The lexical categories 9

4.2 Movement-type diagnostics 83

List of Figures

2.1 A multistratal theory of grammar – the levels of GB theory 8

2.2 Generic $\overline{\mathrm{X}}$-projection for English 10

2.3 The structure of VP 11

2.4 A sample tree 13

2.5 XP adjoined to YP on the left 16

2.6 XP adjoined to YP on the right 16

2.7 An example of a chain 17

2.8 Unrelated chains are allowed 37

2.9 Nested chains of the same type are disallowed 37

2.10 Intersecting chains of the same type are disallowed 38

2.11 Nested chains are permitted in some configurations. 39

3.12 Skeletal $\overline{\mathrm{X}}$ projections created by the chunker: chased = [+past] + chase 49

3.13 The mechanisms of relative clauses 50

4.14 The buffer 59

4.15 The stack 60

4.16 Possible attachment sites 62

4.17 English LF representation 80

4.18 Proposed Chinese LF representation 82

5.19 Possible CP projections 95

Acknowledgements

As everyone claims, there are far too many people to be thanked for their (perhaps unwitting) contributions to this thesis. As is the custom, I mention those I remember at the moment I write this; I would humbly ask those people who I have neglected to mention to forgive my utterly unintentional oversight.

First and foremost, I thank my thesis supervisors, David Poole and Henry Davis, who both forced me to read, think, and write. Without their help and persistence, none of what follows would have been written. I also thank the Department of Computer Science for generous financial support.

I thank David Leblanc for introducing me to Henry, for many a fruitful discussion, and for never being afraid to ask hard questions. For great help with Chinese, I thank Yan Feng Qu. For support, encouragement, and beer, I owe a debt of gratitude to Andrew Csinger and Mike Horsch. For support, encouragement, and coffee, I owe a debt of gratitude to Christopher Romanzin. He provided me with much LaTeX support and thesis encouragement. Thanks also to Scott, Rob, Yggy, Stan, DvanB, Johnny, and the rest of the 312 crowd.

I especially wish to thank my parents. Tack för tillfället att lida (jag skojar) genom skrivandet av en avhandling, och för att kanske ovetande ha lett mig in i den akademiska världen.

Finally, and to me most importantly, my unending thanks to Averill, who has put up with me and my rantings and ravings; who, through it all has stood by me, pushed and prodded me on, reminding me of why I am doing this; and who, despite all this, has committed herself to a lifetime of more of the same.

Chapter 1

Introduction

1.1 Thesis

This dissertation investigates a *psycholinguistically motivated* model of (human) linguistic processing along with the computational and linguistic implications of such a model. The thesis I aim to defend is that there is one parsing mechanism which is sufficient to handle all human languages. A *proof* of this is clearly an unreasonable undertaking; my efforts will therefore be directed towards showing that selected challenges to this hypothesis do not in fact constitute refutations of it.

In order to even begin making claims of universality of the parsing mechanism, the underlying linguistic theory must be cross-linguistically applicable. This has been a driving force behind the development of Government-Binding (GB) theory. GB theory is a constrained theory which makes strong predictions, addresses questions of aquisition, and attains a high level of empirical coverage. It is thus on this theory that the parser is based. A significant obstacle to overcome when basing a parser on a multistratal theory such as GB is being able to recover the appropriate representations at each level of the grammar. This difficulty is made more acute when additional constraints, such as the requirement that the parser be psycholinguistically plausible, are introduced.

It might be argued that the problems of dealing with multiple levels of representation could be easily dispensed with by basing the parser on a monostratal linguistic theory, such as Generalized Phrase Structure Grammar (GPSG) [Gazdar *et al.* 85]. Lexical Functional Grammar (LFG) [Bresnan 82], although a bistratal theory, might also be thought a better framework within which to work — LFG has been developed with computational work in mind. However, neither of these theoretical frameworks can account for the same breadth of linguistic phenomena, in as great a variety of languages, in as constrained (and thus explanatory) a manner, as can GB theory. Furthermore, I show that by using a more suitable parsing model, the multiple levels of GB theory cease to be problematic.

I also aim to achieve an equitable division of labour between competence and performance. I assume that all parameterization to account for cross-linguistic variation is limited to the competence module, and that the performance module is fixed. This is not an uncontroversial assumption. Mazuka & Lust [Mazuka *et al.* 90] claim, for instance, that in order for a "universal" parsing mechanism to be maintained, the parser must be parameterized. By allowing the parsing mechanism to be parameterized, they maintain that there is a family of parsers for natural language. Although I do not directly address the claim of Mazuka & Lust that this parameterization is necessary due to differences in the branching direction of languages (they argue that English must be parsed by a primarily top-down mechanism, while Japanese must be parser using a bottom-up process), I indirectly show that this is not the case, as my parser handles English using a primarily bottom-up approach.

Lee, Chien, Lin, Huang, and Chen [Lee *et al.* 91] describe a parsing mechanism conceived of specifically to handle the syntactic peculiarities of Chinese as compared to English. They imply that Chinese *must* be handled with a mechanism different from that of English. I show that this is not the necessary conclusion. The parsing mechanism I develop in this thesis handles complex constructions in both English and Chinese.

I am not primarily concerned with achieving an extensive linguistic coverage. The main focus is instead on finding ways to deal with theoretically problematic constructions. Because of this, only relevant constructions are dealt with in the main text of the thesis; a greater range of examples are shown in appendix A.2. It is clear that greater linguistic coverage can easily be achieved once the difficult cases are solved.

1.2 Motivation

The field of Artificial Intelligence (AI) encompasses many different activities; I take one of the most important of these activities to be the gaining of a deeper understanding of human intelligence. Constructing systems to carry out tasks which humans do may be useful, but they are of real interest to the field of AI only insofar as they serve as models against which theories of human cognition can be tested.

Rensink & Provan [Rensink *et al.* 91] highlight the relevance of the "naturally intelligent" system in the pursuit of AI:

> Consider a cat in its natural environment. If it is to catch prey and escape from predators, the cat must not only be able to process visual information, but must also do so in real time. Its visual system is therefore best explained not only in terms of limitations on the information available to the eye, but also in terms of limitations on other resources, such as time and space.
>
> There is an increasing awareness — especially within the more computational sub-disciplines of cognitive science — that these more general resource limitations influence many kinds of perceptual and cognitive processes. [pg. 311]

Regardless of whether or not one subscribes to the hypothesis that only humans have the (cognitive) facilities to support a highly structured and productive communication system, it is clear that humans *do* possess such an ability. In constructing artificial systems to process language, heed should be paid to the manner in which humans parse and understand it.

A measure of how closely a model matches reality is needed. In this case, we need to ensure that our model is *psycholinguistically plausible* — that it is consistent with psycholinguistic evidence pertaining to how people process language. Chapter 3 covers a selection of psycholinguistic results which places constraints on the parsing model.

On a more concrete level, part of the motivation for this study is to begin formulating a theory of performance, one which will account for certain linguistic phenomena on its own. By removing the burden of explanation from the competence theory, the competence theory can be simplified; in conjunction, these two simple and orthogonal components function as a more precise and accurate theory.

It should be noted that a theoretically motivated study is not without some practical fallout. A modular and universal parsing scheme will allow construction of natural language applications with a minimum of effort. Once the amount of language-specific code which must be written is minimized, then creating multi-lingual programs is a relatively straightforward task. The syntactic aspects of machine translation should also be simplified if a universal parsing scheme is implemented.

1.3 Outline

The thesis proceeds as follows. The next chapter presents an overview of the linguistic theory in which this work is carried out. It is a short summary of those aspects of Government-Binding theory relevant for the subsequent discussion. Chapter 3 sets up the basic computational framework, describing briefly the parsing models from which I have drawn inspiration, and outlining some relevant psycholinguistic results. Chapter 4 attempts to merge the assumptions of chapters 2 and 3, showing where they do not mesh smoothly, and offering solutions in the form of revisions to either the linguistic or the computational model. Chapter 5 discusses the computational and linguistic implications of this work, while chapter 6 explores some related work. Chapter 7 outlines issues left for future work, and chapter 8 finally concludes the thesis with a summary.

Chapter 2

Linguistic Framework

The role of this chapter is to present a brief overview of the syntactic theory (Government-Binding (GB) theory) underlying this thesis. It is not my intent to give a thorough account of all aspects of GB theory;[1] instead, the focus will be to provide coverage of those areas which are necessary background to understand the remainder of the thesis.

2.1 A Modular Theory

Any theory of natural language syntax must seek to account for the similarities and differences found amongst the languages of the world. Towards that end, GB theory has evolved into a highly modular theory. By isolating various components of the theory from each other, factoring the theory into the lowest or simplest terms, each module becomes simpler, and can hope to yield more accurate predictions. The relevant modules are,

- $\overline{\mathrm{X}}$ Theory
- Case Theory
- θ Theory
- Government Theory
- Control Theory

---

[1] For readers who are interested in exploring GB theory in more detail, there are several good general references, among them [van Riemsdijk *et al.* 86, Lasnik *et al.* 88, Haegeman 91]. For more ambitious readers, [Chomsky *et al.* 92] provides a very good overview of the current status of orthodox GB theory, though it is much more technical and dense than the other references cited.

Each module has associated with it several *parameters* which are set differently for different languages. It is hypothesized that the basic building blocks of human language can be captured in the modules, with slight parametric differences accounting for cross-linguistic variation. As of yet no definitive list of parameters has been forthcoming; some parameters will be mentioned in passing in what follows, to give the reader some feel for how they are meant to function in the general case.

2.2 A Multistratal Theory

Consider the two sentences shown below.

(1) The dog bit the cat.
(2) The cat was bitten by the dog.

The two sentences convey the same information, but with a slightly different emphasis. In the first sentence "the dog" is more prominent, whereas in the second one "the cat" is. We say that "the dog" is the *subject* and "the cat" is the *(direct) object* in (1), while in (2) "the cat" is the *subject* and "the dog" is the *indirect object.* Note that even though the *grammatical functions* of "the dog" and "the cat" are different in the two sentences, "the dog" is doing the biting and it is "the cat" that is being bitten in both. We say that "the dog" is playing the role of the *agent* and "the cat" the role of the *patient*. *Agent* and *patient* are referred to as *thematic roles* (or *θ-roles*). This provides some semantic motivation for thinking of these two sentences as being syntactically related in some manner.

Sentences (1) and (2) are also related morphologically. Active-passive pairs exhibit a predictable morphological variation — *bit* versus *was bitten* in this case. This passive morphology is taken as a further indication that the two could profitably be viewed as being syntactically related in some fashion. This requirement of morphological relatedness excludes the possibility of analysing the following pair of sentences as being *syntactically* related,

(3) John sent the book to Mary.
(4) Mary received the book from John.

Because the sentences in (1) and (2) convey basically the same information and are morphologically related in a predictable fashion, GB theory holds that they are syntactically related in the following way. The two should have a common underlying representation with different surface realizations encoding the difference in attention. Thus were born the levels *D-Structure* and *S-Structure.*[2] D-Structure is the level of representation at which thematic relations are encoded, while S-Structure can be thought of as the interface between two other levels, *Phonological Form* (PF) and *Logical Form* (LF).

In current (mainstream) GB theory, there are four levels in the grammar.[3] D-Structure and S-Structure have survived, while PF and LF are newer additions. PF is meant to deal with post-lexical phonology, and is viewed as the interface to the motor-perceptual system. This level of the grammar will not be discussed further in this thesis. LF is the interface to semantics. It is at this level that things like quantifier scope are represented. This thesis deals with various problems which LF poses for parsing. Figure 2.1 shows the relationships amongst these levels and the *lexicon.*

---

[2] These levels were originally called *Deep Structure* and *Surface Structure.* These labels were abandoned in favour of the abbreviated forms now in use because Deep Structure was perceived as more of a level of semantics rather than as a syntactic level within the model of grammar.

[3] Though in recent work, Chomsky [Chomsky 92] abandons the level of D-Structure.

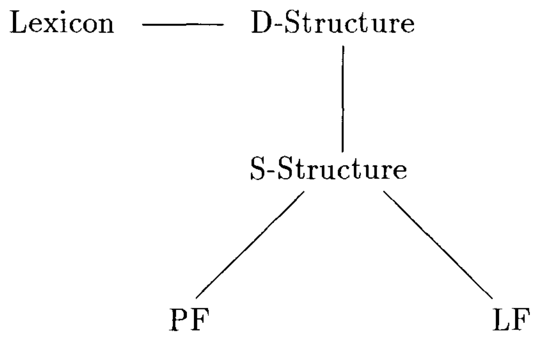


Figure 2.1: A multistratal theory of grammar – the levels of GB theory

The lexicon contains entries for every lexical item. Among other things an item's lexical entry contains its *category.* Many items also include *subcategorization frames.* A verb's subcategorization frame provides information about the types of complements it selects,[4] along with information about any Case or θ-marking which might be assigned to these complements. Prepositions also have subcategorization frames, but most other categories do not.

2.3 $\overline{\text{X}}$ Theory

$\overline{\text{X}}$ (pronounced "x bar") theory is a highly generalized theory of the structure of phrases. As formulated in [Chomsky 86a], the phrasal categories are divided into two different types. The *lexical* categories are based on the features [$\pm$N,$\pm$V], as shown in Table 2.1.[5]

The common abbreviations for these categories are A, V, N, and P. Note that this

---

[4] There are two types of selection, *s-selection* and *c-selection* [Chomsky 86b]. S-selection (or *semantic* selection) refers to the semantic type of the complements that the lexical item takes. C-selection (or *categorial* selection) refers to the grammatical category of a lexical item's complements.

[5] English only has prepositions. Postpositions serve the same purpose as prepositions — they just occur *after* the phrase with which they are associated, rather than before it. To make for easier reading, I will henceforth refer to the class of pre- and post- positions simply as prepositions.

	+N	−N
+V	adjective	verb
−V	noun	pre-/post- positions

Table 2.1: The lexical categories

classification only deals with the *major* lexical categories.

The *non-lexical*[6] categories consist of inflection (abbreviated I), complementizer (abbreviated C), and determiner (abbreviated D). Inflection includes such "items" as tense and agreement; in Chinese, which does not express tense as English does, encodes *aspect* in this position. The complementizer is meant for clausal complementizers, such as "that":

(5) I believe *that* John is lying.

Each of these categories can head a *projection*, according to the following $\overline{\mathrm{X}}$-schema [Chomsky 86a]:[7]

$$\overline{\overline{\mathrm{X}}} \longrightarrow \overline{\overline{\mathrm{Y}}}^{*}\ \overline{\mathrm{X}} \quad , \quad \overline{\overline{\mathrm{Y}}} = \textit{specifier}$$

$$\overline{\mathrm{X}} \longrightarrow \mathrm{X}\ \overline{\overline{\mathrm{Z}}}^{*} \quad , \quad \overline{\overline{\mathrm{Z}}} = \textit{complement}$$

$$\mathrm{X}^{n} \longrightarrow \mathrm{X}^{n}\ \overline{\overline{\mathrm{W}}} \quad , \quad \overline{\overline{\mathrm{W}}} = \textit{adjunct}$$

Let us first take care of some terminology. "X" is known as the *head* of the projection, and is also written X^0. $\overline{\mathrm{X}}$ is also known as the *single bar-level projection (of X)*, and is sometimes written as X'. $\overline{\overline{\mathrm{X}}}$ is called the *maximal projection (of X)*. XP, X'', and $\mathrm{X}^{\mathrm{MAX}}$ are other common notational variants of $\overline{\overline{\mathrm{X}}}$. X^n is the n-th bar-level projection of X, where $n \in \mathbf{N}$.

---

[6] A better term for these categories might be *functional*, though I will use standard terminology throughout.

[7] "*" is the Kleene star, indicating that zero or more occurances of the category are permitted.

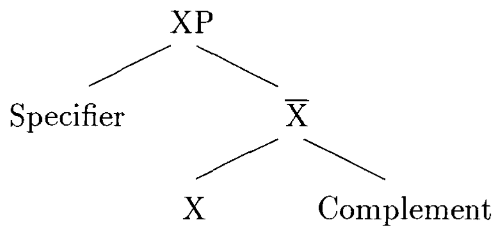


Figure 2.2: Generic $\overline{X}$-projection for English

Note that this schema is not meant to directly encode any *linear* constraints on the order of elements, only *hierarchical* ones.[8] The linear order is determined by various *parameter settings*, which specify, for example, whether specifiers are projection-initial or projection-final, and whether heads are initial or final. Kayne [Kayne 84] argues for a binary-branching structure, which will allow for at most one specifier and at most one complement per projection. Figure 2.2 shows the prototypical $\overline{X}$ projection in English. Note that English has projection-initial specifiers, and that heads precede their complements.

$\overline{X}$ constraints are traditionally satisfied at D-Structure, but need not be satisfied at all other levels of representation.[9]

In recent literature, there have been proposals concerning the structure of VP. Koopman & Sportiche, Kuroda, and Speas [Koopman *et al.* 91, Kuroda 88, Speas 90] argue that the "subject" should be base-generated within the VP. I adopt the Kuroda / Speas version, as shown in figure 2.3, in which the "subject" is generated in the specifier

---

[8] By this I mean that there is no ordering of elements imposed on the right hand side of each individual rule in the schema. This schema does encode linear constraints *indirectly* because crossing branches are not allowed. For example, the specifier could never occur between the head and its complement.

[9] In recent work [Chomsky 92] $\overline{X}$ constraints are satisfied at all levels of the grammar.

of VP position.

For reasons of simplicity, the implementation maintains a uniform structure for all projections, regardless of the category of the head. Using this structure for the VP also, the parser cannot handle multiple-complement structures (as there is only one position available for complements). Though this is clearly undesirable for a general-purpose parser, this limitation is of no consequence for this thesis.

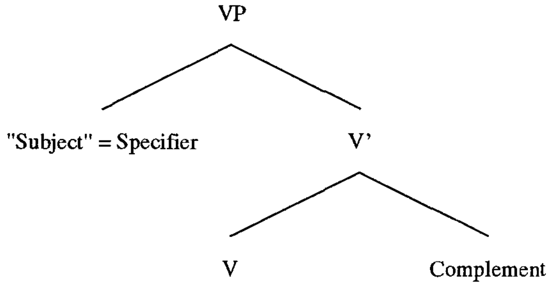


Figure 2.3: The structure of VP

2.4 Basic Definitions

The purpose of this section is to provide a definition of a tree, and to define the basic structural relations between positions in a tree which GB theory makes use of.

2.4.1 Graphs and Trees

The definitions in this section are taken in large part from [Johnsonbaugh 84].

Definition 2.1 *A* graph G *consists of a set* N *of nodes and a set* E *of edges such that each edge* $e \in E$ *is associated with an unordered pair of nodes.*

If the edge e is associated with the nodes m and n, then we write $e = (m, n)$.

Definition 2.2 *Two edges are* parallel *if they are associated with the same pair of nodes.*

Definition 2.3 *An edge associated with a pair of nodes m and n, where $m = n$, is called a* loop.

Definition 2.4 *A* simple graph *is a graph with neither loops nor parallel edges.*

Definition 2.5 *A* path *is a sequence of edges* $\{(n_0, n_1), (n_1, n_2), \ldots, (n_{k-1}, n_k)\}$ *in which the edges are distinct.*

The path shown in the above definition is abbreviated $(n_0, n_1, \ldots, n_k)$.

Definition 2.6 *A* tree *is a simple graph in which (i) there is a unique path between each pair of nodes, and (ii) there is a distinguished node called the* root *of the tree.*

Let T be a rooted tree with root n_0. Suppose that $(n_0, n_1, \ldots, n_k)$ is a path in T. Then we say that (i) n_{k-1} is the mother of n_k; (ii) $n_0, \ldots, n_{k-1}$ are ancestors of n_k; and (iii) n_k is a daughter of n_{k-1}.

2.4.2 Structural Relations

We are now in a position to define some of the basic structural relations used in GB theory. These structural relations are defined over a single tree.

Definition 2.7 α immediately dominates β *iff α is the mother of β.*

For example, in figure 2.4, I′ immediately dominates both I and VP. We can also define the notion of *sisterhood*, which will come to play an important role in θ theory, to be discussed below.

Definition 2.8 α *and* β *are* sisters *(or* are in a sisterhood relation*) if α and β are immediately dominated by the same node.*

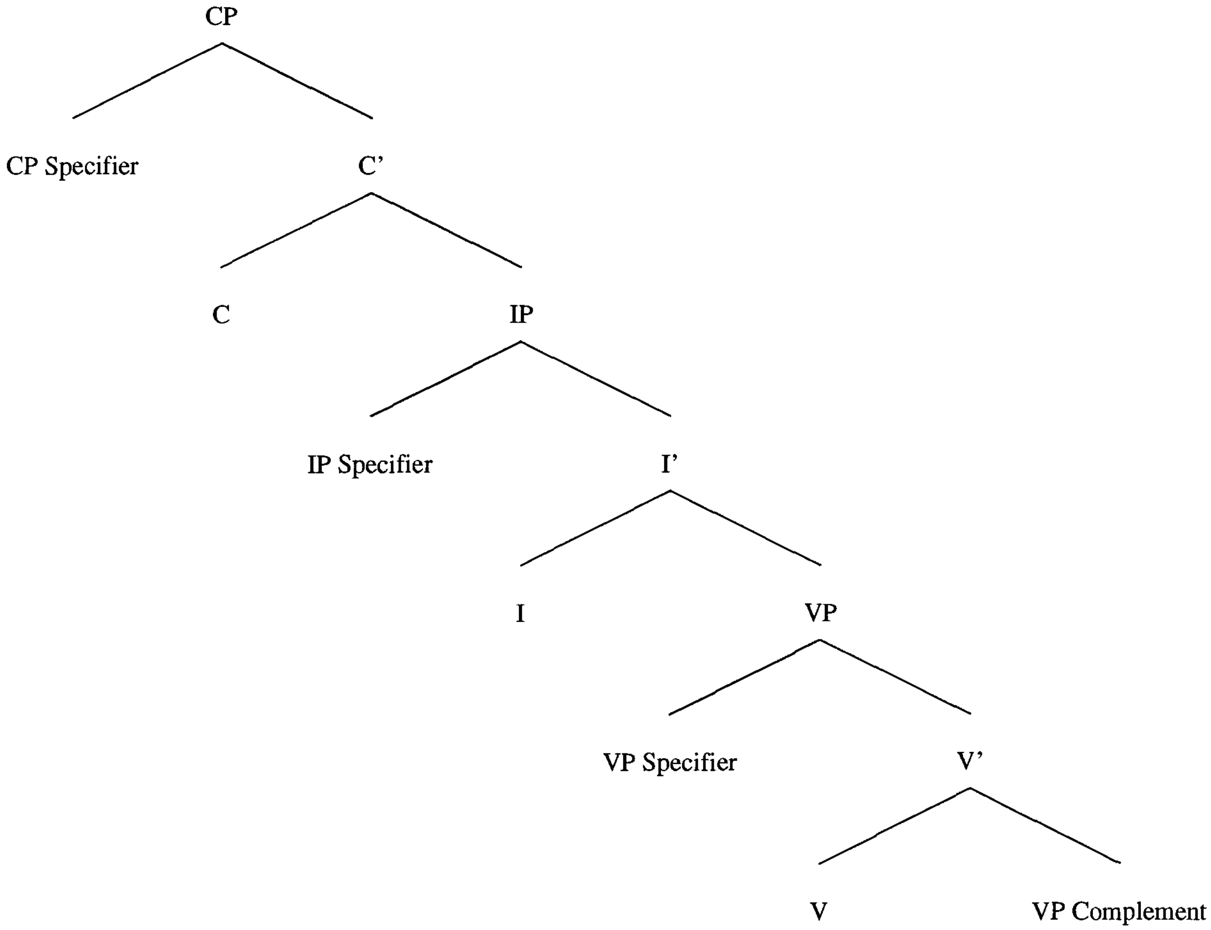


Figure 2.4: A sample tree

For example, I and VP are sisters in figure 2.4, as are IP Specifier and I′.

The relation *dominates* is the transitive closure of the relation *immediately dominates.*

Definition 2.9 α dominates β *iff (i)* α *immediately dominates* β*, or (ii)* α *immediately dominates* γ*, and* γ *dominates* β*.*

In figure 2.4, C′ dominates C and IP, since it immediately dominates them. C′ also dominates everything that either C or IP dominates (namely IP Specifier, I′, I, VP, VP Specifier, V′, V, and VP Complement).

Another very central notion is that of *c-command.*

Definition 2.10 α c-commands β *iff (i)* α *does not dominate* β*, and (ii) every* γ *that dominates* α *dominates* β*.*

Consulting figure 2.4 again, it is easy to verify that I c-commands VP (and vice-versa). I also c-commands VP Specifier, V′, V, and VP Complement. I′, however, only c-commands IP Specifier; I′ does not c-command anything which it also dominates. Finally, note that any node (except the root node of the tree) will c-command itself under this definition.

A closely related notion is that of *m-command*:

Definition 2.11 α m-commands β *iff (i)* α *does not dominate* β*, and (ii) every* γ^{MAX} *that dominates* α *dominates* β*.*

In figure 2.4, consider which nodes I m-commands; I m-commands everything that I c-commands, as well as m-commanding I′ and IP Specifier.

2.5 Affect-α and Chains

The theory provides one mechanism for transforming one level of representation to another. This mechanism is *affect-*α. Crudely, affect-α allows anything to be done, as long as no constraints are violated; some of these well-formedness constraints are explored in the next section. In fact, *affect-*α is a "cover-term" for a number of different processes, the most well-known of which is movement. The movement rule is known as *move-*α; move-α applies only to heads and maximal projections. Under the guise of move-α lurk two distinct operations, *substitution* and *adjunction.*

2.5.1 Substitution

Substitution takes place when an element is moved into a position generated empty in D-Structure. The following examples each exhibit instances of substitution. In each set, the sentence as we would hear it is shown first, while (a) shows the S-Structure representation, and (b) shows the D-Structure representation from which the S-Structure representation is derived. *e* denotes the empty D-Structure position into which the phrase will move at S-Structure. *t* denotes a *trace*. A trace is what is left in place of the phrase being moved — a phonetically empty category which shares the same features as the moved phrase.

(6) *John was tricked.*
a. [John was tricked *t*] (S-Structure)
b. [*e* was tricked John] (D-Structure)

(7) *John seems to be sleeping.*
a. [John seems [*t* to be sleeping]] (S-Structure)
b. [*e* seems [John to be sleeping]] (D-Structure)

(8) *What did Mary give John?*
a. [what did Mary give John *t*] (S-Structure)
b. [*e* Mary gave John what] (D-Structure)

Various constraints act to limit the applicability of substitution. Some of the effects of these constraints are that only heads may move to head positions, and only maximal projections may move to specifier positions.

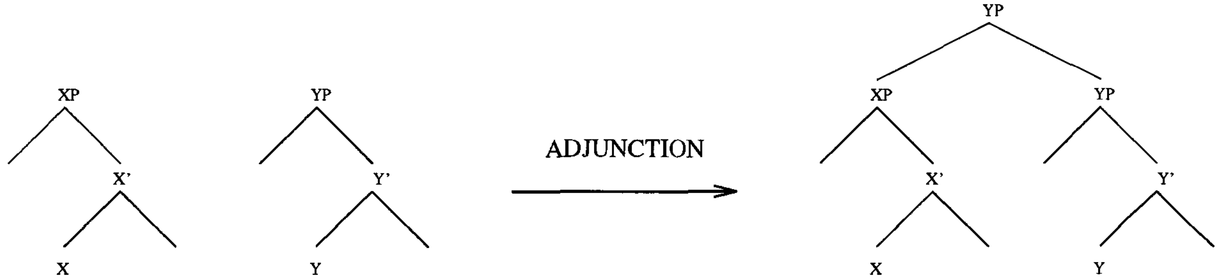


Figure 2.5: XP adjoined to YP on the left

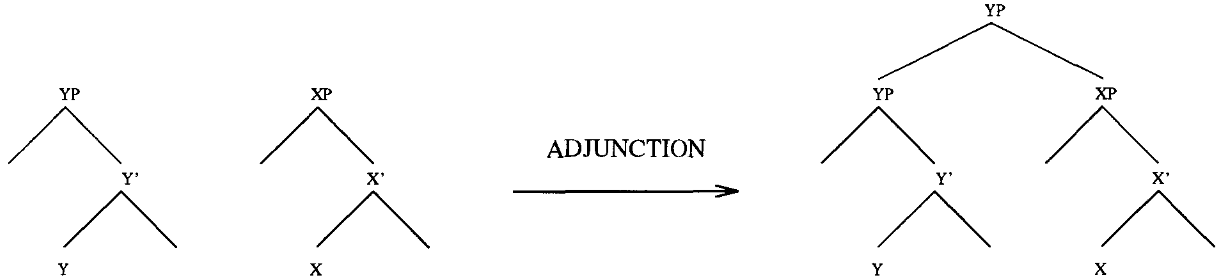


Figure 2.6: XP adjoined to YP on the right

2.5.2 Adjunction

A second manner of incorporating a moved constituent into a tree is called adjunction. Adjunction applies only to maximal projections; a maximal projection XP can be adjoined to another maximal projection YP either on the left of YP or on the right of YP. The adjunction operation makes XP and YP sisters; their mother is a YP. Figures 2.5 and 2.6 show typical cases of adjunction on the left and right, respectively.

2.5.3 Chains

The sequence of traces left by an application of move-α forms a *chain* of traces. Each adjacent pair of traces in a chain is called a *link* of the chain. Links in a chain are subject to locality conditions, defined by the type of the chain. The type of a chain is defined by characteristics of the moved element and the location from which it moved. The chain formed from the movement shown in figure 2.7 is $\mathcal{C} = (\mathrm{XP}, t''', t'', t', t)$. XP is called the

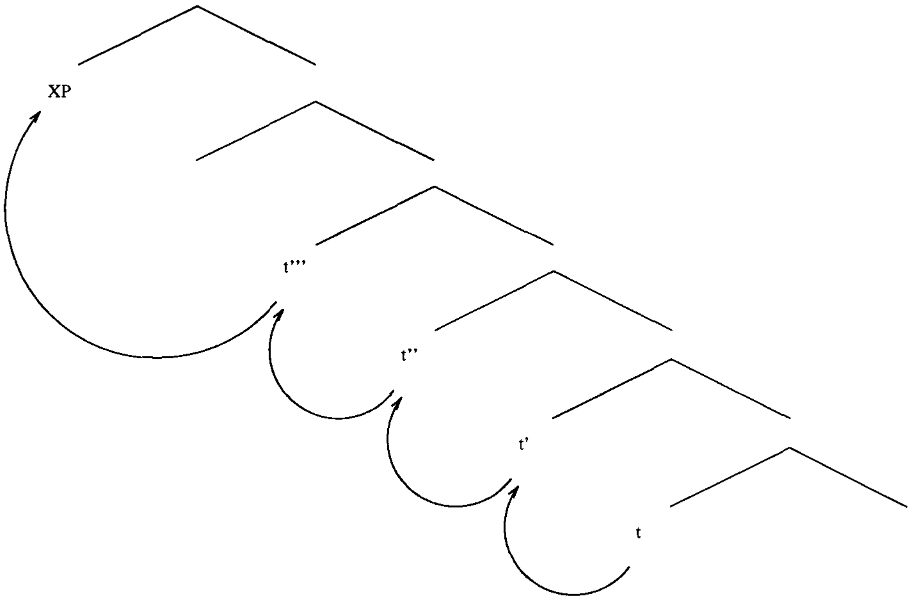


Figure 2.7: An example of a chain

head of the chain, t the *foot of the chain.* A single unmoved element α forms a chain of one element: (α).

What different types of maximal projections are there? According to Rizzi and Cinque [Rizzi 90, Cinque 90] moved elements are distinguished by their degree of referentiality. Their work will be discussed in more detail in section 2.12; it is sufficient to note here that two types of elements are distinguished — those which are highly referential according to some metric, and those which are not.

There are two types of positions from which a maximal projection may be extracted — A- and $\overline{\text{A}}$-positions. A-positions are characterized as those in which θ-roles are potentially assigned. There is disagreement as to the A/$\overline{\text{A}}$ status of certain positions. In recent work [Chomsky *et al.* 92, Chomsky 92, Mahajian 90] an attempt to replace this notion by a more well-defined one, *L-relatedness*, has been made. For the purposes of this thesis an informal, if stipulatory, characterization of the A/$\overline{\text{A}}$ status of positions will suffice. I will consider as A-positions the VP-internal positions (the VP-internal subject position, and the object position) as well as the specifier of IP position. Any other positions I will consider to be $\overline{\text{A}}$-positions.

Heads (X^0s) are not distinguished along the same lines as maximal projections. Movement of heads is strictly local in nature. Since head movement is of little interest as far as this thesis is concerned, it will not be discussed in any detail.

2.6 A-movement

Move-α is constrained by various modules of the grammar. These constraints serve both to limit move-α's applicability in certain situations, as well as to force it in others. *A-movement* is the name given to applications of move-α which are constrained by θ Theory and Case Theory. An *A-chain* is a chain which originates in an A-position.[10]

2.6.1 θ Theory

Sentences (1) and (2) from this chapter, repeated here for convenience, were used to introduce the notion of θ-role.

(9) The dog bit the cat.
(10) The cat was bitten by the dog.

GB theory has one condition to do with θ-roles, the θ-criterion:

θ CRITERION

(i) Every θ-role which a verb assigns must be assigned to an A-chain;
(ii) Every A chain must be assigned exactly one θ-role.

Recalling that even a single unmoved element constitutes a chain, it is clear that in the following examples, the (a) sentences are grammatical, while the (b) sentences

---

[10] Another characterization of an A-chain, not involving the notion of A-position, is one whose foot is in a θ-marked position and whose head is in a Case marked position [Chomsky 86b].

exhibit θ-criterion violations:[11]

(11) a. Carmilla hit the ball.
b. * Carmilla hit.

(12) a. Carmilla gave Franklin the book.
b. * Carmilla kissed Franklin the book.

In the first pair, the (b) sentence violates the first clause of the θ-criterion, while in the latter, the second clause is not met.

2.6.2 Case Theory

English does not exhibit much overt case marking, but what little there is may be seen by considering pronouns.

(13) She gave her the book.

(14) He gave his mother the book.

"She" is marked morphologically with the *nominative* case, while "her" is exhibits the *accusative* case. In the second sentence, "he" has nominative case (just like "she"), and "his" shows *genitive* case marking.

Cross-linguistically, languages show great variation in the amount of overt case marking which is present. GB theory assumes that all languages have case marking, although it may not always be morphologically realized. Case assignment obtains only in very specific situations (the assigner and assignee must be in a sisterhood relation; Case-assignment is also directional), and can therefore instead impose a relatively fixed word order in the language. This is the case in English; for instance, in the sentence,

---

[11] The examples given do not exhibit solely θ-criterion violations — categorial selection, θ-role assignment, and Case marking often go hand in hand; hence the sentences given also violate the Case Filter and selectional (categorial) restrictions.

(15) Mary gave Jane the book.

it is assumed that "Mary" is assigned nominative case and "Jane" accusative case, just like "she" and "her" above, yet there is no overt marking to indicate this. This case marking, since it is not always morphologically realized, is referred to as *abstract* Case (written with a capital C).

Case can be assigned in a number of ways. Verbs assign Case to their complements, as do prepositions. Case can also be assigned to the specifier of IP position; it is assumed that tense features[12] in the I node assign Case to this position.

As important as the positions to which Case is assigned are the positions to which Case is *not* assigned. The VP-internal subject position is not assigned Case by the verb (because this position is, strictly speaking, not a complement of the verb). Since tense features are assumed to assign Case to the specifier of IP position, it follows that in *infinitival* (tenseless) clauses there will be no Case assignment to this position. Passive morphology also is taken to "absorb" Case marking to one of the verb's complements (as well as any "external" θ-marking — see below). The constraint on Case marking can be formulated as follows:[13]

CASE FILTER

Every A-chain must be headed by a Case marked position.

---

[12] This is not the current view of how nominative Case assignment is achieved; other mechanisms have been proposed, mostly for theory-internal reasons. If not entirely correct, using tense features is more than precise enough for my purposes.

[13] There is an exception to this, namely the element PRO (to be discussed in section 2.8). Crucially, PRO must *not* be assigned Case marking.

The following pair can be explained in terms of the Case theory.[14] The (a) version (which also happens to be a D-Structure representation) is ungrammatical because of a violation of the Case Filter; the (b) version avoids a violation of the Case Filter through an application of move-α.

(16) a. * *e* seems John to be nice
b. John seems *t* to be nice

The verb "seem" is a *raising* verb. These types of verbs do not assign θ-roles to their subject positions, but Case is assigned in the main clause by tense features. Since there is no Case assigned to "John" in the embedded clause, application of move-α is forced.

According to Aoun & Li [Aoun *et al.* 89] Chinese does not have any Subject Raising.[15] If there are no Subject Raising phenomena in Chinese, the internal subject hypothesis cannot be maintained. It is therefore assumed that in Chinese, the subject is base-generated in the specifier of IP position, and that it receives the external θ-role of the verb in this position.

Consider now some instances of Case filter violations which do not force movement.

(17) a. * Mary thinks that John to be nice
b. Mary thinks that John is nice

In (a), the embedded clause is infinitival, so there is no Case assignment. The chain $\mathcal{C}$ = (John) does not receive Case marking, and there is no manner in which movement of "John" can satisfy the Case filter. In (b), the embedded clause is tensed, and so Case will be assigned, and the sentence passes the Case filter.

---

[14] The (a) version does not solely violate the Case filter, but also the *Extended Projection Principle*, which is discussed in section 2.7. A sentence such as "* It seems John to be nice" violates only the Case filter. I use the example given in the text to make the D-Structure and S-Structure relationship as transparent as possible.

[15] Though see [Li 90] for a differing view.

Finally, there is a special class of verbs known as *Exceptional Case Marking* (ECM) verbs. These verbs can assign case to the subject of an embedded clause. To illustrate this phenomena, consider the following sentences:

(18) a. Mary thinks he is lying
b. Mary believes he is lying
(19) a. * Mary thinks he to be lying
b. * Mary believes he to be lying

The first pair shows what one would expect, namely that it is acceptable if the subject of the embedded clause, “he”, is assigned Case in the embedded clause. The second pair demonstrates that there is no assignment of nominative Case in the embedded clause, because it is infinitival.

Consider now this pair,

(20) a. * Mary thinks him to be lying
b. Mary believes him to be lying

Here we see a difference between the two verbs. “Believe” is an ECM verb, allowing it to assign case to the subject of the embedded clause. “Him”, though marked with accusative Case, which is the Case marking which objects typically get, is clearly the subject of the embedded clause — “Mary” clearly does not believe “him” (she thinks that he is lying). Another verb which falls into the ECM category is “expect”.

2.6.3 Further Examples of A-movement

Here I present some further examples of A-type movements.

Movement of VP-Internal Subject

The subject of an English clause,[16] although base-generated inside the VP and receiving a θ-role there, cannot remain in this position at S-Structure. The VP-internal subject position is not a Case-marked position, and the subject is thus forced to move in order to pass the Case Filter. If the clause is tensed, then the subject will only move to the specifier of IP position.

(21) a. Tina kissed Kevin.
b. $[_{\text{IP}}\ [_{\text{DP}}\ \text{Tina}\ ]_i\ [_{\text{VP}}\ t_i\ [_{\text{V}'}\ [_{\text{V}}\ \text{kiss}\ ]]]]$

If the clause is untensed, the subject will have to move further up the tree to find a Case-marked position which does not receive a θ-role.

(22) a. Tina seemed to kiss Kevin.
b. $[_{\text{IP}}\ [_{\text{DP}}\ \text{Tina}\ ]_i\ [_{\text{VP}}\ [_{\text{V}'}\ [_{\text{V}}\ \text{seem}\ ]\ [_{\text{IP}}\ [_{\text{VP}}\ t_i\ [_{\text{V}'}\ [_{\text{V}}\ \text{kiss}\ ]]]]]]]$

Passive

The formation of the passive form of an active sentence is also governed by Case and θ considerations. It is claimed that passive morphology (the presence of the auxiliary verb "be" and "-ed" suffix on the main verb in the English case) "absorbs" the external θ-role and one internal Case marking. More formally, passive morphology is taken to change the thematic and Case marking properties of the verb in a consistent manner — the external θ-role is no longer assigned, nor is one internal Case.

The passive sentence,

---

[16] Recall that, according to Aoun & Li [Aoun *et al.* 89], Chinese does not have Subject Raising of any kind.

(23) The cat was chased.

thus has the structure,

(24) $[_{\mathrm{IP}}\ [_{\mathrm{DP}}$ the cat$]_i\ [_{\mathrm{VP}}\ t_i\ [_{\mathrm{V'}}\ [_{\mathrm{V}}$ was $]\ [_{\mathrm{VP}}\ t_i\ [_{\mathrm{V'}}\ [_{\mathrm{V}}$ chased $]\ t_i\ ]]]]]$

In order for the sentence to pass the Case Filter, application of move-α is forced to establish the well-formed chain ($[_{\mathrm{DP}}$ the cat $]_i, t_i, t_i, t_i$).

2.7 Extended Projection Principle

The *Projection Principle* can be stated as follows:[17]

THE PROJECTION PRINCIPLE

The θ-Criterion holds at D-Structure, S-Structure, and LF.

It requires that the thematic relations established at D-Structure be maintained at the other levels of the grammar (save PF). These relations are maintained through chain formation; constraints on the various levels force the application of move-α, while the Projection Principle necessitates the construction of chains.

There are some verbs which do not assign θ-roles to their subjects; the Projection Principle, as it stands, cannot require the presence of a subject in this case. Natural language, however, does seem to require the presence of a subject for each VP. The *Extended Projection Principle* (EPP) requires not only that the Projection Principle be satisfied, but also that every VP have a subject (regardless of whether or not the verb which heads the VP assigns a θ-role to its subject position).

---

[17] This formulation of the Projection Principle is taken from [van Riemsdijk *et al.* 86].

2.8 *pro* and *PRO*

GB theory postulates the existence of a number of different (phonetically) empty elements. It is beyond the scope of this thesis to present the justification for the existence of them (the reader is referred to the various GB overviews cited in the introduction for further reading). Traces have already been introduced; the purpose of this section is simply to mention two other empty elements which will play an important part later in the thesis.

Some languages seem not to require the presence of a subject. In Italian, for example, either of (a) or (b) is a well-formed sentence,

(25) a. Io parlo.
b. Parlo.

"Io" is the pronoun "I", and "parlo" is the first person singular form of the verb "to talk". Both sentences thus mean "I talk". The Extended Projection Principle (EPP) requires that a subject always be present, however. It is thus assumed that a covert subject is present in sentences without an overt one. This empty element is called *pro* (or "little pro"). It is simply a phonetically null pronoun, which may be Case-marked and may receive a θ-role.

The Binding Theory (which also is not discussed in this thesis) deals with possible and required coreference of various elements. A consequence of the Binding Theory, the PRO-Theorem, holds that there is another phonetically empty pronoun, called *PRO* (or "big pro"). PRO must receive a θ-role, but it cannot be Case-marked. A property of PRO is that it must not be governed (see section 2.11), yet Case-marking is taken to occur only in configurations in which government holds. Hence PRO cannot appear in the specifier of IP position of tensed clauses, only in untensed clauses.

2.9 Predication

Predication is a relationship between an antecedent and a predicate. It relates the antecedent to the open position in the predicate. In the following examples[18] a predication relationship holds between the italicized elements,

(26) a. *John* is *sad.*
b. John ate *the meat raw.*
c. *John* ate the meat *nude.*
d. John made *Bill mad.*

Here the predication relation holds between a DP and an Adjective Phrase (AP). The italicized DPs act as antecedents (the thing which is being modified), filling the open position in the AP predicate (the modifier). Considering the preferred interpretations only, it is clear that in the first sentence *sad* is modifying *John.* In the second sentence, it is *the meat* which is understood to be *raw.* In contrast, *John* is taken to be modified by *nude* in the third sentence. In the last example, it is *Bill* who is *mad.*

Each AP above is a *headed* (or *simple*) predicate, so called because the maximal projection of the head is the predicate. Simple predicates can be based upon any of the major lexical categories N, A, P, or V.

There are also *complex* predicates. Such predicates are either IPs or CPs, with an explicit *predicate variable* in the specifier position. A predicate variable defines an open position in the clause, which makes it a one-place predicate. For example, the configuration,

(27) $[_{\text{IP}}$ PRO $[_{\text{VP}} \ldots]]$

---

[18] The discussion and examples of this section derive from [Williams 80].

is a complex predicate, in which PRO is the predicate variable.

Williams construes the predication relation as a coindexation of the predicate and its antecedent. Thus we have, for simple predicates,

(28) a. John$_i$ is sad$_i$

b. John ate [the meat]$_i$ raw$_i$

c. John$_i$ ate the meat nude$_i$

d. John made Bill$_i$ mad$_i$

Some examples of predication involving complex predicates are,

(29) a. John$_i$ promised Bill [$_{\text{IP}}$ PRO to leave]$_i$

b. John persuaded Bill$_i$ [$_{\text{IP}}$ PRO to leave]$_i$

c. John$_i$ tried [$_{\text{IP}}$ PRO to leave]$_i$

These last three examples exhibit what is known as *control.* Of note in these sentences is that the interpretation of the subject of the embedded clause is determined by the verb of the main clause. In the first sentence, John promised Bill that he (John) would leave. In the second sentence, John persuaded Bill that he (Bill) should leave. In the last case, it is John who attempted to leave.

The main clause verb thus "controls" the interpretation of the subject of the embedded clause. The subject of the embedded clause is not overt in any of the cases.

These sentences might at first seem amenable to analysis as A-type movement, as exemplified in,

(30) John$_i$ promised Bill [$_{\text{IP}}$ t_i to leave]

This is not possible, however. The θ-criterion does not allow a DP to receive more than one θ-role, yet if the phenomenon of control were construed as A-movement, then this is exactly what would happen. Consider,

(31) John expected Bill to leave.

"Expect" is an Exceptional Case Marking (ECM) verb, assigning Case to the subject of the embedded clause, but no θ role. John does not expect Bill, he expects Bill to leave. Bill is thus the agent of the embedded verb. In the control case, if this were analyzed as A-movement, then the chain (John$_i$,t_i) would receive two θ roles. t_i is assigned a θ role from "leave", while John$_i$ gets one from "promise". Hence this analysis is not viable.

2.10 $\overline{\textbf{A}}$-movement

$\overline{\text{A}}$-movement is motivated primarily by scope considerations of the moved operator and selectional properties of verbs. Not all types of $\overline{\text{A}}$-movement are obligatory; topicalization, for instance, is an optional stylistic movement. In this section *wh*-movement constructions and infinitival relative clauses, both of which are directly relevant to the discussion in chapter 4, will be considered. Appendix A.2 contains many further examples of $\overline{\text{A}}$-movements and the parse trees which the parser produces for them.

2.10.1 *Wh*-movement

Wh-movement is an $\overline{\text{A}}$-type movement dealing with the movement of question words from their base-generated positions to the positions they need to occupy at LF. *Wh*-movement is exemplified in the following examples,

(32) a. Who dislikes Susan?
b. What did Susan buy?

The structures of these sentences are,

(33) a. $[_{\text{CP}}$ Who$_i$ $[_{\text{C}'}$ $[_{\text{C}}$] $[_{\text{IP}}$ t_i $[_{\text{I}'}$ $[_{\text{I}}$] $[_{\text{VP}}$ t_i $[_{\text{V}'}$ $[_{\text{V}}$ dislikes] Susan]]]]]]

b. $[_{\text{CP}}$ What$_i$ $[_{\text{C}'}$ $[_{\text{C}}$ did] $[_{\text{IP}}$ Susan $[_{\text{I}'}$ $[_{\text{I}}$] $[_{\text{VP}}$ buy t_i]]]]]

A question word[19] such as "who" or "what", is moved to the front of the clause, and an $\bar{\text{A}}$-chain is established between the *wh*-phrase and the trace.

In English, *wh*-movement takes place between D-Structure and S-Structure (this is *overt* movement). In some languages, such as Chinese, *wh*-phrases appear in their D-Structure positions at S-Structure,

(34) *Ni da le shui*

you hit ASP who

"Who did you hit?"

At first there might seem to be little if no justification for postulating that Chinese has *wh*-movement. Huang [Huang 82] investigated Chinese *wh*-constructions, and found motivation for proposing that Chinese has *wh*-movement between S-Structure and LF (*covert* movement).

Move-α is subject to a number of constraints. These movement constraints explain why *wh*-phrases cannot be extracted from certain constructions. Thus, while (a) is fine, (b) and (c) are both deviant to some degree. Moreover, under the intended interpretation,[20] (c) is significantly worse than (b).

---

[19] Question words are referred to as *wh*-words since they, in English, generally begin with the letters "wh" (as in "who", "what", "when", and "where"; "how" is an exception); these "words" are actually considered to be complete phrases (DPs), and so they are more often referred to as *wh*-phrases. This terminology, though *not* cross-linguistically applicable, is used throughout the literature.

[20] The point is that "why" cannot be interpreted as question about Marty's liking; "why" can only refer to Fred's belief.

(35) a. Who$_i$ does Fred believe [$_{\text{CP}}$ that Marty likes t_i]

b. ? Who$_i$ does Fred believe [$_{\text{NP}}$ the claim [$_{\text{CP}}$ that Marty likes t_i]]

c. * Why$_i$ does Fred believe [$_{\text{NP}}$ the claim [$_{\text{CP}}$ that Marty likes Beverley t_i]]

If Chinese does not have covert *wh*-movement, then no such asymmetries should be exhibited; conversely, if these asymmetries are found in Chinese, they can be taken to be indicative of movement. Indeed, these asymmetries do exist in Chinese,

(36) *Fred xiangxin Marty xihuan shui?*

Fred believe Marty like ASP who

a. *"Who does Fred believe that Marty likes?"*

(37) *? Fred xiangxin Marty xihuan shui de shuofa?*

Fred believe Marty like ASP who COMP claim

a. *"Who does Fred believe the claim that Marty likes?"*

(38) ** Fred xiangxin Marty weishenme xihuan Beverley de shuofa?*

Fred believe Marty why like ASP Beverley COMP claim

a. *"Why does Fred believe the claim that Marty likes Beverley?"*

Further support for the hypothesis that Chinese has *wh*-movement comes from selectional restrictions, discussed below in section 2.10.3.

2.10.2 Operator Scope

GB theory holds that an operator[21] must, at some level of representation, make its scope explicit by structural means. This is done through the c-command relation — whatever

---

[21] According to Chomsky [Chomsky 81] operators include, among other elements, quantifiers and *wh*-phrases.

subtree an operator c-commands at the relevant level of representation is taken to be its scope. An operator will bind a variable which falls in its scope. For example, when a *wh*-phrase is moved, as in,

(39) What$_i$ did Averill buy t_i

the interpretation is,

(40) What is the thing x, such that Averill bought x

What$_i$ binds the $\overline{\text{A}}$-trace t_i, a variable. The interpretation of the trace is dependent on the operator.

There is a prohibition against the occurrence of unbound (or free) variables in natural language. There is also a prohibition against vacuous quantification; an operator, in order to be licensed to appear in a phrase structure tree, must bind some variable. In general, an operator may bind only one variable.

Languages differ as to the level of representation at which the scope of operators is made explicit. In Polish and Hungarian, scope is represented mostly at S-Structure; in English, scope is represented at S-Structure for *wh*-phrases, and at LF for quantifiers, while in Chinese, scope is represented at LF for *wh*-phrases, and at S-Structure for quantifiers.

2.10.3 Selectional Restrictions

Certain verbs require that their embedded clause either have or lack the *wh* feature; they select a clause that is either [+*wh*] or [−*wh*]. The paradigm below demonstrates this,

(41) a. Beverley asked me who bought books.
b. * Who does Beverley ask me bought books?

(42) a. * Beverley believes who bought books.

b. Who does Beverley believe bought books?

(43) a. Beverley knows who bought books.

b. Who does Beverley knows bought books?

The verb "wonder" selects for a [+*wh*] clause, while "think" selects a [−*wh*] clause. The verb "know", on the other hand, selects both. This same contrast is found in Chinese, even though there is no overt *wh*-movement going on which could move the *wh*-phrase into the required position, the embedded clause CP specifier.

(44) *Zhangsan wen wo shui mai le shu.*

Zhangsan ask I who buy ASP book

a. *"Zhangsan asked me who bought books."*

b. * *"Who did Zhangsan asked me bought books?"*

(45) *Zhangsan xiangxin shui mai le shu*

Zhangsan believe who buy ASP book

a. * *"Zhangsan believes who bought books."*

b. *"Who does Zhangsan believe bought books?"*

(46) *Zhangsan zhidao shui mai le shu*

Zhangsan know who buy ASP book

a. *"Zhangsan knows who bought books."*

b. *"Who does Zhangsan know bought books?"*

Data such as this provides additional support for postulating LF *wh*-movement for Chinese. Assuming that Chinese (and similar languages) have covert *wh*-movement allows significant generalizations to be captured. However, as will become evident in chapter 4, covert movement also causes problems for a psycholinguistically plausible parser.

2.10.4 Relative Clauses

A relative clause is a clause which modifies a DP. As such, the relative clause is a one-place predicate. In the example below, the relative clause is "who Mary despises"; it modifies the DP "the man".

(47) Mike saw the man who Mary despises

Just like an AP, a relative clause is a one-place predicate, the antecedent of which is identified through predication. However, unlike the case of an AP, the antecedent of a relative clause must occur in a specific structural configuration. Thus, the predicate is adjoined to and coindexed with its antecedent:

(48) $[_{\text{DP}}$ *antecedent* $[_{\text{CP}} \ldots]]$

The complete structure of the above example is thus,

(49) Mike saw $[_{\text{DP}} [_{\text{DP}}$ the man $]_i [_{\text{CP}}$ who$_i$ $[_{\text{IP}}$ Mary despises t_i $]]_i$ $]$

There is an $\bar{\text{A}}$-chain which exists between who$_i$ and t_i, while a predication relationship holds between $[_{\text{DP}}$ the man $]_i$ and the predicate; the predicate variable who$_i$ bears the same index as the clause and the antecedent.[22]

Empty Operator Constructions

The relative clauses considered so far have all been tensed (or *finite*) relative clauses. Being tensed, their specifier of IP positions are Case-marked. There are also untensed

---

[22] According to Browning [Browning 87] there are two mechanisms necessary to make the proper link between the predicate and the predicate variable. These mechanisms are *feature percolation*, whereby the head of a phrase agrees in all its features with its maximal projection, and *SPEC-HEAD agreement*, whereby a head agrees with its specifier. For the purposes of this thesis I choose to ignore these additional mechanisms in the interests of simplicity of implementation. A "faithful" parser should, of course, implement these relations as well; at present, a direct coindexation between the antecedent and the predicate variable is carried out. See section 3.1.4 for more details on the implementation.

(or *infinitival*) relative clauses, which do not have Case-marked IP specifiers. Infinitival relative clauses are a type of *empty operator* construction. Empty operators are simply operators which are phonetically null.

Browning [Browning 87] discusses many different types of empty operator constructions, among them *purpose clauses.* Since purpose clauses and infinitival relative clauses actually have the same structure, I will discuss primarily the latter.[23]

Infinitival relative clauses can be of either the subject-gap or the object-gap variety,

(50) a. Jennifer is the woman to watch Marlene. (subject-gap)
b. Jennifer is the woman to watch. (object-gap)

In the first sentence, "Jennifer" is going to watch Marlene. In the second sentence, "Jennifer" is perceived as the woman that should be watched, by some arbitrary person. There clearly is a difference in interpretation, which can only be resolved by the end of the sentence. The following pair shows that the "resolution point" might be embedded arbitrarily from from the start of the relative clause ([the woman ...]).

(51) a. Jennifer is the woman to tell the police to watch Marlene.
b. Jennifer is the woman to tell the police to watch.

---

[23] Purpose clauses have internal structures associated with them identical to those associated with infinitival relative clauses. The difference between the two types of clauses is that the purpose clause interpretation is possible only with a restricted set of verbs. The purpose clause reading can in these cases be forced by including the phrase "in order", as follows,

John bought the bike (in order) to ride.

The two possible interpretations are as follows. The first is the relative clause reading, while the second is the purpose clause reading.

John bought "the bike", the one which it is really cool to ride.
John bought the bike in order to ride every day to and from school.

Since I am interested in the difficulties posed by the difference in structure between subject gap and object gap varieties of these clauses, I choose to simply consider infinitival relative clauses.

Although the subject-gap / object-gap pairs seem to be of equal parsing complexity, their commonly accepted structural analyses would predict that they should be of very different processing complexity. Browning proposes the following structures for subject-gap and object-gap purpose clauses (and also for infinitival relative clauses).

(52) a. $[_{\text{IP}}$ PRO $[_{\text{I}'}$ $]]$
b. $[_{\text{CP}}$ pro_i $[_{\text{C}'}$ C_i $[_{\text{IP}}$ PRO $[_{\text{VP}} \ldots t_i$ $]]]]$

The first is a simple predication structure, in which PRO is the predicate variable. The second structure also involves predication. The operator binds a trace, which is interpreted with respect to the antecedent, while PRO receives an arbitrary interpretation.

Browning notes that the major difference between subject-gap and object-gap purpose clauses is the incompatibility of the subject-gap variety with dative shift structures. She argues that thematic orientation rather than a structural difference accounts for this. Substantial structural differences thus seem ill motivated; subject-gap purpose clauses thus act more as a subclass of object-gap purpose clauses than as a class on their own.

2.11 Government

There are two types of *government*, a structural relationship, which are important in chain formation. The first is *head government*, the second *antecedent government*; these are defined as follows (Relativized Minimality and the notion of barrier will be discussed in the next section),

Definition 2.12 α head-governs β *iff (i)* $\alpha \in \{A, N, P, V, I\}$*; (ii)* α *m-commands* β*; (iii) no barrier intervenes; (iv) Relativized Minimality is respected.*

It must be assumed that I does not head-govern its specifier position when it does not contain tense features.

Definition 2.13 α X antecedent governs β *($X \in \{A, \overline{A}, X^0\}$) iff (i) α and β are coindexed; (ii) α c-commands β; (iii) no barrier intervenes; (iv) Relativized Minimality is respected.*

2.12 Relativized Minimality

Chomsky [Chomsky 86a] introduces a condition of *minimality* into government,

Definition 2.14 *In the configuration* $\ldots\alpha\ldots[_\gamma \ldots\delta\ldots\beta\ldots]$, *$\gamma$ is a barrier for β if γ is the immediate projection of δ, a zero-level category distinct from β.*

Rizzi [Rizzi 90] notes that Chomsky's minimality condition is *rigid* in that only if γ is a head does its presence interfere with government by α; furthermore such a head γ prevents both head government *and* antecedent government by α. Rizzi thus proposes that minimality should be *relativized* according to the type of government. This implies that a local potential head governor will prevent a distant head from governing into its domain, and that a local antecedent governor will prevent a distant antecedent governor from the same; however, potential head governors interfere in no way with antecedent government, and likewise, potential antecedent governors have no effect on government by heads.

Taking this concept of relativity a step further, Rizzi also proposes to limit interaction of potential antecedent governors to those of the same type: X^0, A, or $\overline{\text{A}}$. Relativized Minimality (RM) thus construed restricts the formation of multiple chains of the same type. Figure 2.8 shows that completely disjoint chains are allowed under RM.

As a first approximation, nested chains of the same type are not permitted under RM. Figures 2.9 and 2.10 depict configurations which are not permissible if the chains are of the same type — nested chains and overlapping chains respectively.

RM is, however, stated in terms of *typical potential* governors for some element.

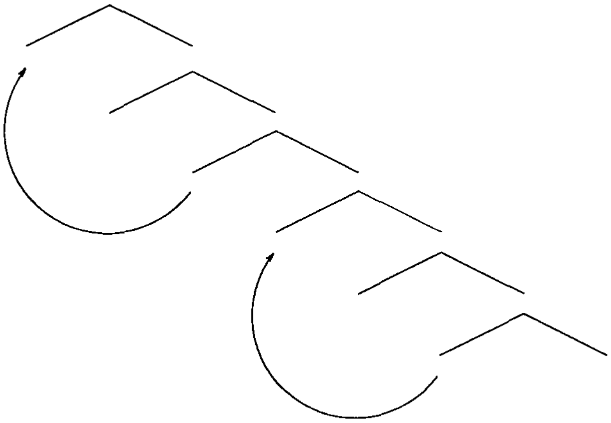


Figure 2.8: Unrelated chains are allowed

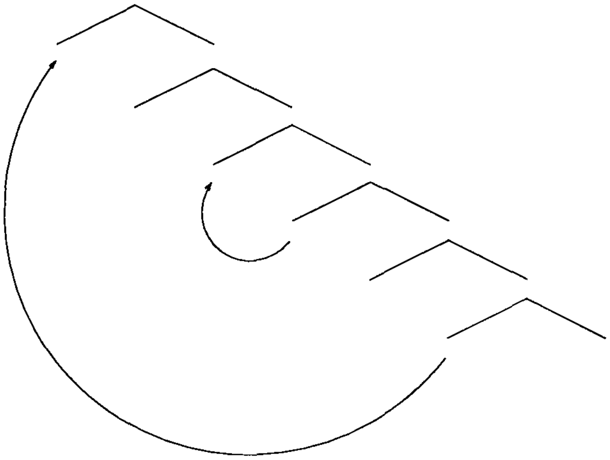


Figure 2.9: Nested chains of the same type are disallowed

RELATIVIZED MINIMALITY

> *α X-governs β only if there is no γ such that (i) γ is a typical potential X-governor for β; (ii) γ c-commands β and does not c-command α.*

A typical potential governor is any element of the proper type which could, in some configuration, actually be a governor.

Definition 2.15 *γ is a* typical potential head governor *for β iff γ is a head m-commanding β.*

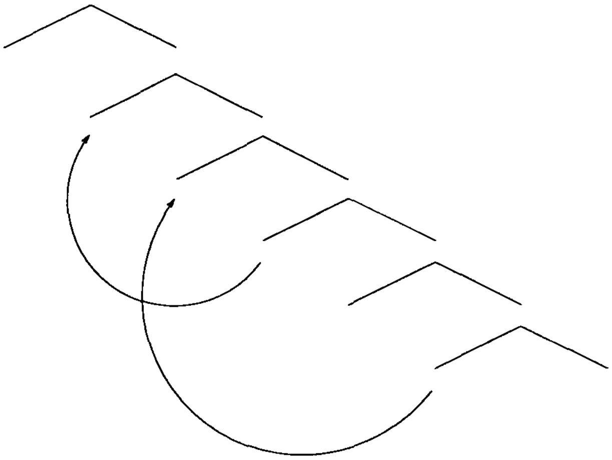


Figure 2.10: Intersecting chains of the same type are disallowed

Definition 2.16 *γ is a* typical potential A antecedent governor *for β iff γ is an A specifier c-commanding β.*

Definition 2.17 *γ is a* typical potential $\overline{\text{A}}$ antecedent governor *for β iff γ is an $\overline{A}$ specifier c-commanding β.*

Definition 2.18 *γ is a* typical potential X^0 antecedent governor *for β iff γ is a head c-commanding β.*

If the possible government domain of neither chain interferes with that of the other, nesting of chains is allowed. This situation obtains when the embedded chain is contained in a left branch (in a right-branching language) of the tree, as in figure 2.11.

In order to account for the paradigm seen in (53) – (59), both Rizzi and Cinque [Rizzi 90, Cinque 90] appeal to the degree of referentiality of the moved phrase. Rizzi ties a phrase's degree of referentiality to the θ-role it receives, whereas Cinque calls upon Pesetsky's [Pesetsky 87] notion of D-linking. I will leave open the question of what an appropriate definition of referentiality might be, since no one notion has been settled upon (nor are any of the proposals particularly well-defined at the moment).

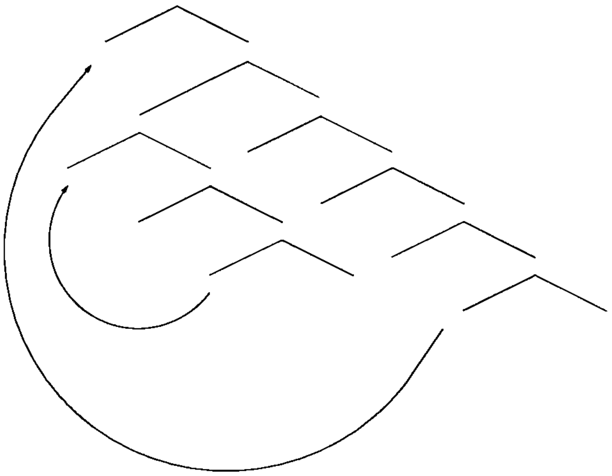


Figure 2.11: Nested chains are permitted in some configurations.

Cinque allows for two types of $\overline{\text{A}}$-dependencies. Antecedent government is available to those elements which are not "referential enough"; antecedent government is a relatively restricted type of dependency-establishing method. Binding is the other manner in which Cinque allows an $\overline{\text{A}}$-dependency to be set up. Binding is much less restricted than antecedent government.

Definition 2.19 *α* binds *β iff (i) α and β have the same referential index; (ii) α c-commands β.*

(Antecedent) government chains are sensitive to government barriers,

Definition 2.20 *Every maximal projection that fails to be directly selected by a category nondistinct from* [+V] *is a* barrier for government.

while binding chains are blocked by binding barriers,

Definition 2.21 *Every maximal projection that fails to be (directly or indirectly) selected in the canonical direction by a category nondistinct from* [+V] *is a* barrier for binding.

The canonical direction is the direction in which a verb assigns Case-marking to its complement. For English this is rightward. The categories non-distinct from [+V] are

A, V, I, and C. Finally, the different notions of selection are defined as,

Definition 2.22 α directly selects β, *where* α *is a lexical category, iff* α *directly s-selects* β.

Definition 2.23 α directly selects β, *where* α *is a non-lexical category, iff* α *directly c-selects* β.

Direct in these cases refers to *direct* θ-marking (for internal θ-roles), as opposed indirect θ-marking (for external θ-roles).

2.13 The Empty Category Principle

It has long been observed that there exists an asymmetry between the extraction of arguments versus adjuncts from various domains. Historically, domains from which extraction could not take place are called *islands* (see [Ross 67]). Two classes of islands are recognized: weak islands and strong island. It is the weak islands which allow the argument/adjunct asymmetry to be seen. In the following examples, taken from Cinque [Cinque 90] (his (1) – (7)), the first three are considered strong islands, while the last four are weak islands.

(53) *Subject island*

a. * Which books did [talking about *t*] become difficult?

b. * How would [to behave *t*] be inappropriate?

(54) *Complex NP island*

a. * To whom have you found [someone [who would speak *t*]]?

b. * How have you found [someone [who would fix it *t*]]?

(55) *Adjunct island*

a. * To whom did you leave [without speaking *t*]?

b. * How was he fired [after behaving *t*]?

(56) Wh-*island*

a. ?? To whom didn't they know [when to give their present *t*]?

b. * How did they ask you [who behaved *t*]?

(57) *Inner (negative) island*

a. To whom didn't you speak *t*?

b. * How didn't you behave *t*?

(58) *Factive island*

a. To whom do you regret [that you could not speak *t*]?

b. * How do you regret [that you behaved *t*]?

(59) *Extraposition Island*

a. To whom is it time to speak *t*?

b. * How is it time to behave *t*?

The *Empty Category Principle* (ECP) was formulated to account for this paradigm. It is a principle which has appeared in numerous forms in the literature [Lasnik *et al.* 84, Chomsky 86a, Kayne 84, Aoun *et al.* 87]. I adopt Cinque's formulation, which derives from that of Rizzi [Rizzi 90].

THE EMPTY CATEGORY PRINCIPLE

> *A nonpronominal empty category must be properly head-governed by a head non-distinct from* [+V].

Proper head-government is defined as follows,

Definition 2.24 α properly head-governs β *iff* α *head-governs* β *and* α *c-commands* β.

2.14 Minimal Linguistic Assumptions

This chapter has presented a short overview of current syntactic theory within the GB framework. Although the problems to be discussed are cast in terms of GB theory, as are the solutions proposed, the ideas behind them are not tied to this theoretical model. Indeed, there is a minimal set of syntactic assumptions, as well as a minimal set of processing assumptions, which, when taken together, yield the problems which this thesis deals with. This section presents such a minimal set of syntactic assumptions, which are (relatively) uncontroversial.

It is assumed that there is some level of syntactic description at which the quantificational structure of sentences is reflected. In GB theory, this is the level of logical form (LF).

It is further assumed that the scope of a *wh*-phrase (or an operator, in general) is represented structurally at this level of representation. In GB theory, *wh*-phrases must c-command their scope at LF; they come to c-command their scope through application of move-α (this subcase commonly being referred to as *wh*-movement).

Finally it is assumed that this process of identifying the scope of a *wh*-phrase may occur either overtly or covertly. In GB theory, this implies that *wh*-movement may take place either between D-Structure and S-Structure (overt movement), or between S-Structure and LF (covert movement).

Chapter 3

Computational Framework

This chapter sets up the computational framework within which the work of this thesis is carried out. The general parsing model is presented first, alongside an exploration of various models on which it is based. Some minimal computational assumptions are outlined in the second section. The third and final section of the chapter discusses some motivating psycholinguistic evidence for these assumptions.

3.1 A General Model

This section considers various proposals regarding psycholinguistically plausible parsing methods. I will discuss each in turn, finally summarizing the general parsing model which I will use. Further refinements to this general model will be described in the next chapter, as the motivation for them are encountered and explored.

There are certain facts about the human language processing apparatus which a psycholinguistically plausible model of parsing must be consistent with; minimally, it must recognize that,

- humans have quite strict short-term memory limitations, and,
- humans process linguistic input very quickly, and,
- not all sentence constructions are of equal processing complexity.

3.1.1 The Marcus Parser

Marcus [Marcus 80] made the first serious attempt to formulate a psycholinguistically plausible parsing mechanism. Marcus made several very important observations about how such a model must be structured.

Most importantly, Marcus claimed that in order to be psycholinguistically plausible, a parsing mechanism must be *deterministic*. Marcus presented this idea as the *Determinism Hypothesis*:

> ...the syntax of any natural language can be parsed by a mechanism which operates "strictly deterministically" in that it does not simulate a nondeterministic machine ... [pg. 2]

Deterministic parsing entails that the parser must exhibit certain behaviours. First of all, Marcus observes that it must be the case that any structure which is built is *permanent* — the parser cannot be allowed to backtrack:

> In terms of the structures that the interpreter creates and manipulates, this will mean that once a parse node is created, it cannot be destroyed; that once a node is labeled with a given grammatical feature, that feature cannot be removed; and that once one node is attached to another node as its daughter, that attachment cannot be broken. [pg. 12]

Furthermore, no structure which is built may ever be discarded:

> ...all syntactic substructures created by the grammar interpreter for a given input must be output as part of the syntactic structure assigned to that input. [pg. 12]

Finally, the internal state of the parser must not be able to encode temporary syntactic structure. If this were the case, then the previous conditions would be vacuous.

In addition to exhibiting the above behaviours, the parser must, in order to be deterministic, operate in an at least partially *bottom-up* fashion. This implies that it will be driven by the input stream, projecting structure from the lexical items. It cannot operate exclusively bottom-up, however. The parser must "be able to reflect expectations that follow from general grammatical properties of the partial structures built up during the parsing process." [pg. 14] For example, subcategorization requirements are a form of *top-down* parsing. More generally, *selection* by one category of some particular category involves expectation on the parser's part of what it should "see" in the input stream. For example, C obligatorily selects IP as a complement, just as I obligatorily selects VP as a complement. (See section 3.1.2 for a more detailed discussion of selection.)

It is also assumed that the parser processes input in a strictly left-to-right manner. Although a parser would have to take as input a sound stream in order to truly claim to model the human language processing system, current technology prevents this. A parser taking written input, however, must certainly process it in the same order as a human would hear it if it were spoken, if any claims of psycholinguistic plausibility are to made.

The left-to-right order processing constraint, together with the determinism hypothesis, entails that the parser must have access to a certain amount of *lookahead.* This allows the parser to postpone deciding what to do with a certain element in the input stream until it has seen some number of further elements from the input stream. One can think of this as being a "lag" in processing. The amount of lookahead must be constrained, else the determinism claim is invalidated — by using unbounded lookahead the parser would, in effect, be voiding the determinism claim.

The Marcus parser utilizes two major data structures. The first is a buffer, a first-in first-out data structure. The buffer contains nodes which are seeking mother nodes. The second is a stack, a last-in first-out data structure. The stack contains nodes which are seeking daughters. In brief, the parser reads input from the input stream, places partially processed input in the buffer, and tree fragments in the stack.

3.1.2 Licensing Parsers

There has been interest recently in methods of building phrase structure trees without using extensive phrase structure rules. This section describes two such parsers; although what I adopt is closer to the work of [Abney *et al.* 86] than that of [LeBlanc 90], there are certain aspects of the latter which deserve mention here.

Abney

Abney and Abney & Cole [Abney 87, Abney *et al.* 86] describe how *licensing conditions* can be used to build phrase structure. Every element in a phrase structure representation must serve some purpose by being there; something must license its presence. As Abney [Abney 87] puts it, "Specifically, every element in the structure is licensed by performing a particular function in the structure; the structure is well-formed only if every element in it is licensed." [pg. 2]

The licensing conditions which Abney [Abney 87] uses are (i) functional selection, (ii) subjecthood, (iii) modification, and (iv) θ-role assignment.

Abney assumes that the functional selection relationship holds between C and IP, between I and VP, and between D and NP. IP is the only possible complement of C, as are VP and NP of I and D respectively. This information is very useful to the parser when it is building phrase structure. Functional selection, though similar to c-selection, is more restrictive. V c-selects (and s-selects) its complements, but V does not functionally select them.

The subjecthood relation is intended to license a subject's appearance; it is basically an encoding of the Extended Projection Principle's requirement that every clause have a subject.

Modification is intended to license modifiers. Since I do not deal with modifiers, I will not discuss this condition further.

The θ-criterion is a very strong condition, requiring both that every A-chain have exactly one θ-role, and that every θ-role be assigned to exactly one A-chain. Abney makes use of this, using θ-role assignment as a licensing condition. In other words, one way in which a maximal projection is licensed to appear in a phrase structure tree is by being assigned a θ-role.

Abney also assumes that sisterhood is necessary in order for a licensing relationship to hold. A directionality parameter is introduced into each licensing condition, allowing cross-linguistic variation to be accounted for.

I believe that the use of licensing conditions to build phrase structure is correct, though I do not adopt the same set of licensing conditions as Abney. I outline the licensing relations I incorporate into my model below.

LeBlanc

LeBlanc [LeBlanc 90] builds phrase structure without using an explicit $\overline{\text{X}}$ schema. Based on a revision of GB theory by Davis [Davis 87], various *Percolation Principles* are used to derive the categorial features of a dominating nodes given two adjacent nodes.

LeBlanc's parser is driven by Case marking and θ-role saturation. In his use of Case marking to build phrase structure, LeBlanc's parser differs from that of Abney. Case marking is primitive to GB theory, and together with θ-assignment, it is central to A-movement. The adoption of Case marking as a licensing mechanism thus seems natural.

3.1.3 Chunk Parsing

In recent work Abney [Abney 91] has further explored the construction of phrase structure representations. It is observed that humans seem to parse linguistic input in "chunks".

Abney proposes a *chunking parser* as more psycholinguistically plausible than one which deals directly with the lexical items. Following the spirit, though not the letter, of Abney's proposal, the parser herein described projects each lexical item to its full $\overline{\mathrm{X}}$ projection. These skeletal $\overline{\mathrm{X}}$ projections are then joined together by the parser proper (this is what Abney calls the "attacher").

There are three exceptions to this simplistic chunking procedure. The first involves the construction of DPs. LeBlanc notes that noun phrases (be they conceived of as NPs or DPs) must be handled in special ways by the parser. I simply enforce the functional selection constraint as early as possible, namely in the "chunker" portion of the parser, to ensure that a complete DP is the smallest chunk that will be returned after encountering a determiner or a noun. Modifying prepositional phrases (PPs) can cause difficulties, as there may be arbitrarily many of them. Nominalizations such as,

(60) the destruction of the city

involve subcategorization. Although this subcategorization is optional, the chunker can use such information as a cue to search for a potential complement. If a complement is found, it is incorporated into the structure, else the structure is closed off at this point. Cases of adjuncts, which are not subcategorized for, are not dealt with.

The second special case is that of a tensed verb, which is analyzed as tense features and an untensed verb. Each of these projects a skeletal $\overline{\mathrm{X}}$ projection. Figure 3.12 shows how the inflected verb "chased" is analyzed by the chunker.[1]

The chunker also handles the case of head movement of an auxiliary in a special manner. For further details, refer to section 4.2.

Abney also notes that the chunker can contain language-specific rules. Although

---

[1] Chinese does not have tense; instead, the parser recovers aspectual features from the verb, which head the I projection. The I projection is thus parameterized for either "tense" or "aspect".

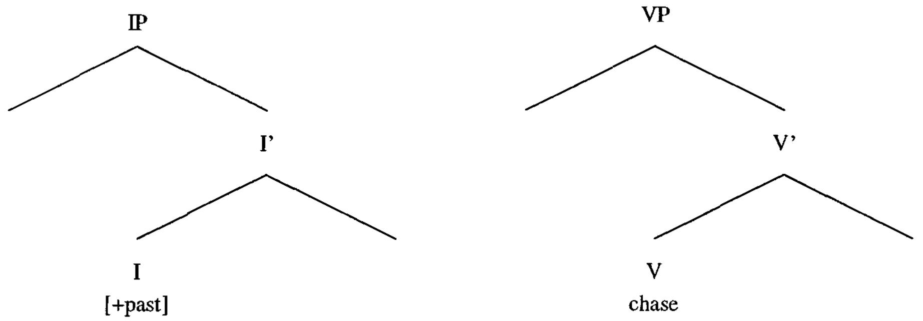


Figure 3.12: Skeletal $\overline{\mathrm{X}}$ projections created by the chunker: chased = [+past] + chase

I do not think this is necessary, for coding simplicity I utilize two slightly different chunking procedures to handle the difference between English, in which "tense" heads the I projection, and Chinese, in which "aspect" heads the I projection.

3.1.4 Other Details

As mentioned in a footnote in section 2.10.1, not all mechanisms prescribed by the theory as necessary to represent relative clauses correctly are implemented in the parser. In order to simplify the programming involved, the mechanisms of feature percolation and SPEC-HEAD agreement are not implemented. Furthermore, the antecedent of the relative clause is not coindexed with the relative clause. This would result in a representation such as,

(61) $[_{\text{DP}}$ $[_{\text{DP}}$ the man $]_i$ $[_{\text{CP}}$ who$_i$ $[_{\text{IP}} \ldots t_i \ldots]]_i]$

with the relationships shown in figure 3.13. Rather, the antecedent is coindexed with the predicate variable, which in turn binds some variable in the relative clause. When the parser identifies that a predication relationship needs to be established, it directly coindexes the predicate variable and the antecedent. In other words, the parser produces a structure similar to the proper one, but in a more direct and *ad hoc* manner.

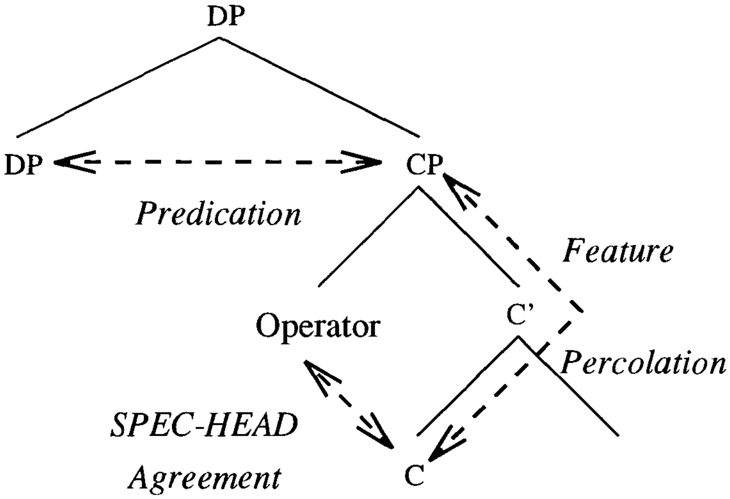


Figure 3.13: The mechanisms of relative clauses

(62) $[_{\text{DP}}\ [_{\text{DP}}$ the man $]_i\ [_{\text{CP}}$ who$_i\ [_{\text{IP}} \ldots t_i \ldots]]]$

The relationship between $[_{\text{DP}}$ the man $]_i$ and who$_i$ is taken to be that of predication.

As will be further discussed in Chapter 4, the implementation does not handle head movement very well. Since the primary purpose of the implementation is to demonstrate how $\overline{\text{A}}$-movements are handled by the parsing scheme, head movement is carried out in a fairly *ad hoc* manner at present. Proper handling of head movement is an issue left for future work.

A further weakness of the implementation is that adjuncts are not handled at all. There are several interesting aspects to adjuncts; though they do not bear directly on the points being made in the thesis, a natural extension of this work must address how they are parsed. Parsers driven by licensing mechanisms in general have a difficult time with adjuncts, since the appearance of adjuncts does not seem to be governed by any licensing conditions.

3.1.5 Summary

To conclude this section, I will summarize the parsing model which I assume as a starting point for this thesis.

The parser has two major (conceptual) components, corresponding to Abney's chunker and attacher. It is of the traditional buffer and stack variety, as defined in [Marcus 80]. The chunker takes lexical items from the input stream, constructs skeletal phrase structure representations (chunks), and deposits them at the end of the buffer. The attacher uses the buffer and the stack; it has direct access to the first element of the buffer and the top element of the stack (although it can look at all the items in the buffer, it can only modify the first one). It shuffles nodes between the two, employing the licensing mechanisms to attach nodes to each other to eventually form a complete phrase structure tree.

The licensing mechanisms I use are:

- θ-role assignment
- Case-marking
- functional selection
- scope assignment
- prohibition against vacuous quantification
- Extended Projection Principle (EPP)

The first three licensing mechanisms were addressed above. The remaining ones, though discussed in chapter 2, deserve a brief second mention here. The scope assignment requirement is used to license the appearance of quantificational elements and to flag movement of such elements. The prohibition against vacuous quantification forces the presence of a variable for a quantificational element to bind. In other words, if a quantifier has been identified, the appearance of a variable which this quantifier can bind is not only licensed, it is required. Finally, the EPP forces the presence of a subject in each clause. The EPP thus licenses the appearance of pleonastics, as in

(63) It seems Tina hit Kevin.

In conjunction with the prohibition against vacuous quantification, I use the EPP to solve a parsing dilemma in infinitival relative clause constructions. In these cases, a the presence of a PRO subject is required.

3.2 Minimal Computational Assumptions

In this section the basic assumptions about the parsing model are laid out. These assumptions, which are relatively uncontroversial, taken in conjunction with the linguistic assumptions presented earlier, conspire to present seemingly insurmountable difficulties for *any* parsing mechanism adhering to them.

I assume that the parser has a limited lookahead capability. Though there is no explicit limit on the amount of lookahead the parser may use built into it, no more than two chunks' worth of material is looked at before a decision is reached as to what to do with the current item.

The parser does not backtrack. Once a piece of phrase structure has been attached to another, the parser is committed and cannot undo this attachment.

I also assume that the parser operates in a strictly *left to right* fashion. This simply implies that the parser is constrained to process items from the input stream in the same order in which a person would hear them.

It is further assumed that the parser operates primarily bottom up (with subcategorization and functional selection accounting for the top-down aspects).

Finally, I assume that the parser is filler-driven rather than gap-driven in its chain-construction process (see below).

3.3 Psycholinguistic Evidence

The purpose of this section is to present psycholinguistic evidence for the processing assumptions outlined at the end of the previous section.

3.3.1 Filler-Driven Parsing

Frazier & Flores-D'Arcais [Frazier *et al.* 89] studied gap locating strategies in English and Dutch. Their study concluded that the then commonly held view that parsing was *gap-driven* could not be correct. Parsing is gap-driven if the parser makes no attempt to establish a dependency between a filler and a gap until the gap is identified; in terms of GB theory, chain construction would in this case be triggered by the identification of the gap. Instead they advocate the use of a filler-driven strategy. Chain construction is in this case triggered by the identification of a filler. Kurtzman, Crawford, & Nychis-Florence [Kurtzman *et al.* 91] consider primarily how *wh*-traces are located; their results also support a filler-driven approach.

3.3.2 General Parsing Principles

Frazier & Rayner [Frazier *et al.* 88] conducted experiments in which subjects read sentences while their eye movements were tracked and recorded. The measurements allowed both the global and local complexity which the sentences presented to the subjects to be gauged. Frazier & Rayner proposed a number of constraints to account for their results. Among those constraints are the following,

Left-to-Right Constraint This constraint ensures that the parser will process words in the order which it encounters them (i.e.: from left to right).

First Analysis Constraint The parser pursues only one possible analysis at one time, and does not attempt multiple analyses in parallel.

Late Closure The parser should attempt to attach each new chunk into the most recently constructed parse tree fragment.

Most Recent Filler When constructing multiple chains, the most recently identified filler should be associated with the first gap found.

These constraints each support the computational assumptions I make. Though not formulated as an explicit constraint, the fact that people have a very limited short-term memory capacity is an underlying motivation for the First Analysis Constraint, Late Closure, and the Most Recent Filler constraint. My assumption of limited lookahead is a more explicit expression of this restriction.

The Left-to-Right Constraint certainly supports the assumption that the parser operates in a left to right manner.

The First Analysis Constraint indirectly justifies the non-backtracking assumption. As Marcus [Marcus 80] notes, if backtracking is allowed, then the determinism claim is invalidated, and this is equivalent to pursuing multiple parses at once.

Late Closure is implied by the operation of the stack and buffer of the parser, as the attacher only works with the first element of the buffer and the top element of the stack.

The Most Recent Filler strategy, as will be seen in the next chapter, is incorporated very neatly into the parsing mechanism, and is used to explain certain linguistic effects.

Chapter 4

Problems and Solutions

The computational and linguistic assumptions made in the previous two chapters seem innocuous enough, yet they interact to produce seemingly insurmountable difficulties. This chapter studies a selection of syntactic constructs and linguistic effects, and the processing difficulties they pose given these assumptions. Since my main focus is on dealing with theoretically problematic and therefore interesting phenomena, rather than striving simply to achieve a substantial empirical coverage, only a limited number of pertinent examples are considered.

Three main issues are dealt with in this chapter. I first consider Relativized Minimality (RM), a set of conditions on possible chain-interactions. I argue that RM is better analyzed as a processing (performance) constraint rather than as a linguistic (competence) constraint. The second issue concerns overt as opposed to covert movement; the difficulties posed by cross-linguistic variation in the syntactic level at which $\overline{\text{A}}$-movement takes place are studied. Finally, the parsing problems caused by infinitival relative clauses (in English) of various types are looked at. There are two broad classes of infinitival relative clauses of interest, so-called subject-gap and object-gap clauses. One of these two classes is predicted to be unparsable given the previously outlined assumptions, while they are both equally easy to process in reality.

The majority of problems are caused by $\overline{\text{A}}$-type movements, and hence attention is focussed on them. However, A-type movements are considered as well; the parsing mechanism is refined throughout the chapter, and straightforward examples of the parser's operation make the rest of the chapter more easily understood.

4.1 Relativized Minimality Effects

The Marcus parser has a serious limitation, in that it cannot handle multiple *wh*-movement structures. Consider, for example,

(64) Which problem did John wonder how Mary solved?

The Marcus parser is able to keep track of only one *wh*-dependency, and only one which can be analyzed as arising from successive cyclic movement. Recalling *Relativized Minimality* (RM) and the related discussion from Chapter 2, it becomes apparent that some mechanism more sophisticated than that used by the Marcus parser is needed to handle various filler-gap dependencies. Moreover, as both Rizzi and Cinque [Rizzi 90, Cinque 90] observe, there are certain long distance $\overline{\text{A}}$-dependencies which *cannot* be analyzed as arising from successive cyclic *wh*-movement.

To summarize, RM captures the generalization that there are (at least) three distinct forms of dependencies induced by move-α — (i) $\overline{\text{A}}$ chains, (ii) A chains, and (iii) X^0chains (or head movement chains). Chains of different types do not interfere with each other, but in the case of chains of the same type, there must be no closer potential antecedent governor for the foot of the chain.[1] This type of constraint on chains appears to be amenable to analysis as a parsing constraint rather than a grammar constraint, in the following way.[2]

---

[1] Thanks to Mark Baker for pointing out that the effect of this is that only in left-branching contexts (in a predominantly right-branching language like English) do multiple dependencies of the same type occur. Thus a single store, rather than a a stack, is required to achieve the desired effect.

[2] There are two ways of explaining linguistic effects in terms of parsing constraints. The first is to encode the appropriate constraints directly into the parsing mechanism; this is the route I follow (this is a *direct* manner of explanation). The second casts the relevent constraints as part of the syntax, but holds that they exist solely to satisfy processing limitations (an *indirect* account). The first approach

The parser must have some way of keeping track of the different chains which it is constructing, and must be able to keep separate the three different types of chains. The behaviour prescribed by RM is achieved if the parser has a separate stack[3] for each type of chain it is constructing.

It is not necessary for the parser to keep track of the whole chain, but merely the current *link* in the chain. Once a link has been successfully created, it is clear that there is no interference from other potential antecedent governors. Keeping only one link of the chain in memory at once lessens the memory load on the parser — a requirement if the model is to maintain claims of psycholinguistic plausibility. If the model necessitated keeping the whole chain in memory, the model would require a potentially unbounded amount of memory, as chains (especially $\overline{\text{A}}$ chains) are potentially unbounded.

The parser must also be able to differentiate between the two different types of $\overline{\text{A}}$ dependencies which may be established [Cinque 90], *antecedent-government* and *binding.* A constraint on the required degree of referentiality of the moved element is not yet well defined in the literature [Pesetsky 87, Rizzi 90, Cinque 90]. Since $\overline{\text{A}}$ dependencies are of much greater interest in this thesis than the partitioning of lexical items into various classes, I have chosen to simply encode directly in the lexicon the type of $\overline{\text{A}}$ dependency each lexical item enters into. Once a proper characterization of this constraint emerges it can be straightforwardly incorporated into the parsing mechanism, and the current stipulative approach dispensed with.

There are also reasons why RM should not be considered a purely syntactic (competence) constraint. As it is defined, it is not open to cross-linguistic variation. If the parser is universal cross-linguistically, while the competence theory is parameterized, then the

---

results in a more constrained theory. It allows only two types of constraints — processing constraints and syntactic constraints. The latter approach also allows processing-derived syntactic constraints, as well as syntactically-derived processing constraints.

[3] Though, as noted, only in left-branching structures may other elements be pushed onto the stack; else the parser operates as though it has a single store for moved elements of a certain type.

strong hypothesis is that *all* universal principles[4] belong to the performance theory and *only* cross-linguistically variable principles are part of the competence theory. Although this is most certainly too strong a statement, RM seems to be well-motivated as a parsing constraint in the first place. Furthermore, at least as far as *nested* chains are concerned, there is no reason why the competence theory should rule this out, as ... $filler_1$ ... $filler_2$ ... gap_2 ... gap_1 ... structures have easily identifiable filler-gap relationships. However, there is a very good performance-related reason as to why nested structures are ruled out — they require a large amount of short-term memory to resolve properly. Moreover, RM violations, such as *wh*-island constraint violations, do not result in completely deviant structures. Instead, there seems to be a gradient effect. This is generally taken to be an indication of a performace rather than a competence constraint.[5] Thus, as a performance constraint, RM has a good explanation; as a competence constraint, it seems singularly odd.

4.2 Brief Overview of the Parser's Operation

In this section I present a short discussion of the data structures used, as well as an outline of how the parser operates.

The main data structures of the parser are a buffer, a main stack, and several movement stacks. The buffer is a First-In First-Out (FIFO) structure (see figure 4.14). Elements enter the buffer from the rear, and leave it from the front. The parser tries to always have at least two elements in the buffer, but it will not allow more than four elements in the buffer at once.

The main stack is a Last-In First-Out (LIFO) structure (see figure 4.15). Ideally,

---

[4] I take a *universal* principle to be one which applies cross-linguitically without change. In other words, I use universal to mean unparameterized in this case.

[5] The parser as implemented does not provide graded judgements; if a violation of RM occurs, the parser simply stops the parse at that point.

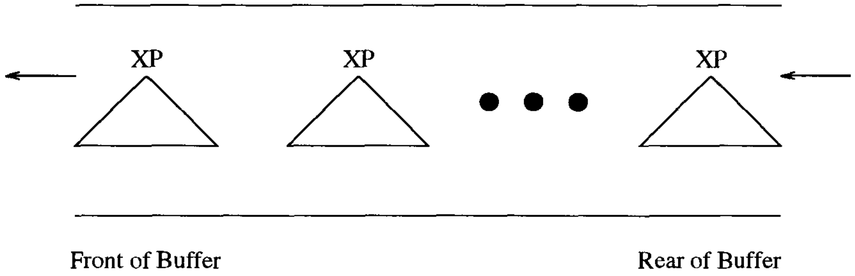


Figure 4.14: The buffer

the stack should have a limited size as well, but using Marcus' model and assumptions, this is not possible (see section 3.1.1). The parser is not able to connect phrase structure fragments together as soon as it should. Instead, it is forced to wait until a phrase is *complete* before attaching it to something else. A projection is complete when all of its selectional restrictions have been met; this includes functional as well as lexical selection. Lexical selection includes the selection of complements specified in subcategorization frames. The reason that attachment cannot be carried out until a projection is complete is that once two pieces of phrase structure are connected, the parser cannot access the internal structure of the newly formed phrase in order to attach complements to a piece of it. This leads to a very large number of tree fragments sitting on the stack, which is not psycholinguistically plausible (see section 3.3.2 and chapter 7 for relevant discussion).

As an example, when a tensed verb is processed, both an IP chunk and a VP chunk are constructed (see section 3.1.3 for details). The VP chunk cannot be connected to the IP chunk until all of its selectional restrictions are satisfied. Until then, both the IP and the VP must just sit on the stack. If these two tree fragments are part of an embedded clause, the problem is compounded, as the IP cannot be connected to the higher-level clause until complete; it will not be complete until the VP is complete.[6]

---

[6] A question which leaps to mind is how potentially unbounded structure, such as a DP with stacked adjectives, might be handled. Although the parser at present does not deal with such constructions, a

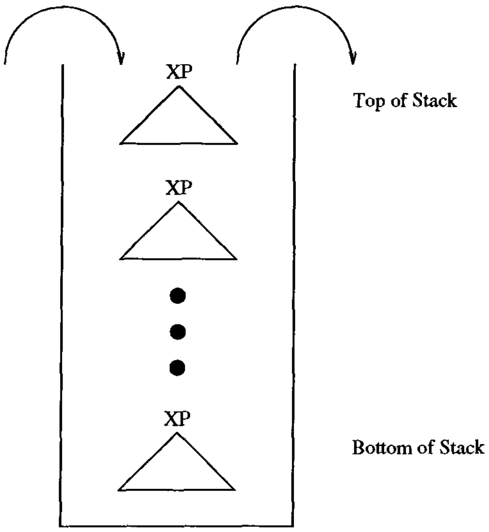


Figure 4.15: The stack

The parser makes use of three auxiliary data structures for moved elements. These data structures are stacks, though pushing of new elements is restricted to left branching structures, due to the linear nature of the parse.

The parser proceeds by cycling through a sequence of possible operations, performing the first one which is possible in the current configuration, and then beginning a new cycle. The possible operations may be grouped as follows:

Head Movement Head movement is handled by the parser only to the extent that it must, and in an *ad hoc* manner. The single case of head movement the parser recognizes is that of an auxiliary moved from I to C.

The chunker recognizes configurations in which an element of category I cannot occupy the head position of an IP. These configurations are (i) a *wh*-phrase followed

---

proper chunking procedure can be used to parse them without the need for an unbounded data structure to store the iterating elements. When the chunker enounters a determiner it recognizes that an NP must be a part of the final DP which the determiner will head. If, in the course of searching for the noun which heads this NP the chunker encounters a sequence of stacked adjectives, it can simply incorporate them into the partially built DP, without storing them in a buffer or stack. If new items are continually incorporated into existing structure, no increased processing load results.

by an auxiliary, and (ii) an auxiliary followed by a DP. In these cases, a skeletal CP is projected, with the head of the CP occupied by the auxiliary of category I.

The parser assumes that a projection headed by an element of the wrong category was moved there via head movement. It thus begins to construct an X^0-chain, placing an appropriate trace in the head movement stack. The head movement chain is terminated when the landing site (an empty head of the proper category) is found, subject to Relativized Minimality.

Replenish Buffer The parser tries to keep at least two elements in the buffer at all times. To satisfy this need, the chunker is invoked. The chunker processes items from the input stream, generating chunks which are placed in the buffer. The chunker does this either until there are at least two chunks in the buffer, or until there are no more items in the input stream.

Predication The conditions under which the predication relation holds are checked next. If licensed, the proper configuration is constructed and the proper relationships are established between the relevant elements (as discussed in section 3.1.4).

Attachment This is the main part of the parser; it is the embodiment of the attacher (see sections 3.1.3 and 3.1.5). There are two main types of attachment. The first involves the attachment of two pieces of phrase structure to each other, while the second involves the incorporation of a trace into a piece of phrase structure. The parser always attempts to do these two types of attachment in that order.

The parser always considers the element located at the front of the buffer and the element at the top of the stack when attempting to attach two phrase structure fragments to each other. These are the only two positions of these data structures which are accessible to the attacher.

In order to maintain the linear order of items from the input stream, the attacher

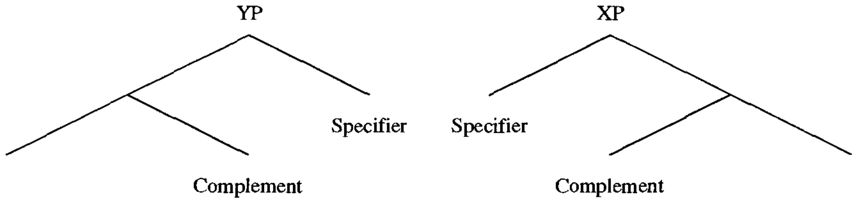


Figure 4.16: Possible attachment sites

has four choices as to how to try to attach these fragments together. Referring to figure 4.16, the possible attachments are as follows (where XP is sitting at the front of the buffer, and YP is located at the top of the stack):

1. If the specifier of XP is on the left, an attempt to attach YP into this position is made.
2. If the complement of XP is on the left, an attempt to attach YP into this position is made.
3. If the complement of YP is on the right, an attempt to attach XP into this position is made.
4. If the specifier of YP is on the right, an attempt to attach XP into this position is made.

If any attachment is possible, the first one (according to the order given) is carried out.

Concerning the incorporation of a trace into a phrase structure representation, the parser considers the same attachment sites, in the same order, for any trace which might be present in one of the movement stacks.

CP Insertion The conditions under which a CP should be inserted (such as when a *wh*-phrase has been identified, but there is no overt complementizer present) are tested; if needed, a skeletal CP is dropped into the first position of the buffer.

Stopping Conditions The conditions under which a parse is successful are checked. These conditions are (i) the input stream is empty, (ii) the buffer is empty, (iii) there is a single phrase structure representation in the stack, and (iv) all the movement stacks are empty.

Extended Projection Principle The constraints of the Extended Projection Principle are checked (see sections 2.7 and 2.8), and if necessary complied with at this point.

Else If nothing else can be done, the parser attempts to remove an element from the buffer and push it onto the stack. If the buffer is empty, then this will not be possible — in this case the top element of the stack is popped and placed at the front of the buffer.

4.3 A-type Movements

A-type movement is driven by Case and θ-role saturation, and is thus relatively easy for the parser to "undo" in the course of building up the parse tree. The Case filter and the θ criterion together conspire to force various DPs to move from a θ-marked position at D-Structure so as to occupy a position where Case is assigned by S-Structure.

This means that the "undoing" of A-type movement will be triggered by the identification of a DP in a Case marked but non-θ marked position. The parser begins building a chain for this moved element, and must therefore keep track of a "trace" of it.

The parser will deposit a trace of this moved element in each compatible position it encounters, until the terminal landing site has been encountered. Any non-Case marked position, whether θ marked or not, is a possible landing site. If the position is not θ marked, it is an intermediate landing site, while if it is θ marked, it is (necessarily) the foot of the chain.

Once a suitable landing site for the trace has been found, it will be inserted into the tree structure at this point. A local check of the Case and θ marking of the trace is done to see if all the well-formedness conditions on chains are met at this point. If the trace has been assigned both Case marking and a θ role, then the A-chain is complete. The store for A-type movement is therefore cleared of its contents. If the Case and θ role requirements of the trace have not yet been met, the contents of the A-movement stack remain, and the current chain is thus continued.

Let us now consider a detailed example of how the parser actually handles a simple A-type dependency. Take an ordinary English sentence such as,

(65) The dog chased the cat.

This sentence is analyzed as having the structure shown below,

(66) $[_{\text{IP}}\ [_{\text{NP}}$ the dog$]_i\ [_{\text{I}'}\ [_{\text{I}}\ ]]\ [_{\text{VP}}\ t_i$ chased the cat $]]$

At first the buffer, (main) stack, and movement stacks are all empty, while the input stream contains the words of the sentence to be parsed. The parser has only limited access to the input stream; the chunker can remove the first element from the input stream, and project a skeletal $\overline{\text{X}}$ structure for it (recall section 3.1.3). The initial state can thus be represented as follows, where "–" serves to separate visually the various elements in the buffer and the stack.[7] Recall also that the buffer is a First-In-First-Out (FIFO) structure, while the stack is a Last-In-First-Out (LIFO) construct. In the parser state snapshots shown below, items enter the buffer from the right and leave it from the left, while items enter and leave the stack from the left (i.e.: the top of the stack is on the left).

---

[7] As this example deals with an A-type movement, only the relevent movement stack is shown in these "snapshots"; the other movement stacks remain empty throughout the parse.

Store	Contents
Input	the dog chased the cat
Buffer	
Stack	
A mvnt stack	

As described in section 3.1.5 the chunker operates by projecting skeletal $\overline{\mathrm{X}}$ structure from heads, and placing these chunks in the buffer. Three special cases discussed were (i) the parsing of DPs, (ii) the projection of both a skeletal IP and VP from a tensed verb, and (iii) the processing of the auxiliary verb "do". Thus, at this point, since a DP always functionally selects an NP, the chunker parses the first two elements from the input stream, *the* and *dog*, as a chunk, a DP. This complete DP is placed in the buffer.

Store	Contents
Input	chased the cat
Buffer	$[_{\mathrm{DP}}$ the dog$]$
Stack	
A mvnt stack	

The parser attempts to always have at least two chunks in the buffer at once; it uses this lookahead to avoid backtracking. Whenever there are fewer than two chunks in the buffer, the chunker is invoked to process items from the input stream until there are two or more chunks in the buffer. Since the processing of one item from the input stream may result in more than one chunk (consider the case of a tensed verb, which results in two chunks, a skeletal IP projection and a skeletal VP projection), there may be more than two chunks in the buffer after the chunker finishes its operation. Once there are at least two chunks in the buffer, the chunker relinquishes control to the attacher.

Briefly, the parser's basic order of operation involves four phases. The first priority for the parser is to keep at least two chunks in the buffer, if possible. This will not be possible if the input stream is empty. The second phase involves trying to attach pieces of phrase structure to each other, while the third one is chain construction. Finally, the last phase, which is entered only if nothing else can be done, shuffles material between the buffer and the stack.

Let us now return to the example. At this point in the parse there is now only one chunk in the buffer, and so the chunker is activated to replenish the buffer. This next word, the tensed verb *chased*, is extracted from the input stream and analyzed as [+past] and *chase*. The chunker recognizes that the feature [+past] must head an IP, and *chase* a VP; it builds skeletal $\overline{X}$ projections headed by these elements, and places them in the buffer. Although IP functionally selects a VP, these two chunks cannot yet be attached, as the VP is not *complete*.

Store	Contents
Input	the cat
Buffer	$[_{\mathrm{DP}}\ \text{the dog}] - [_{\mathrm{IP}}\ [_{\mathrm{I'}}\ [_{\mathrm{I}}\ +\text{past}]]] - [_{\mathrm{VP}}\ [_{\mathrm{V'}}\ [_{\mathrm{V}}\ \text{chase}]]]$
Stack	
A mvnt stack	

The parser now has at least two chunks in the buffer — in fact, there are three. The attacher works with the first element of the buffer and the top element of the stack, trying to put them together somehow. Since the stack is empty, and no attachment can thus be done, the parser removes the first element from the buffer, and pushes it onto the top of the stack.

Store	Contents
Input	the cat
Buffer	$[_{\text{IP}}\ [_{\text{I}'}\ [_{\text{I}}$ +past]]] – $[_{\text{VP}}\ [_{\text{V}'}\ [_{\text{V}}$ chase]]]
Stack	$[_{\text{DP}}$ the dog]
A mvnt stack	

The DP at the top of the stack needs to be assigned both Case and a θ-role. Since there is Case to be assigned in the specifier of IP position, the DP is licensed to appear there. The attacher carries out the attachment; the structure resulting from any attachment is always placed at the front of the buffer. The attacher also recognizes that the θ requirements of the DP have not been satisfied. This, as described above, is the trigger for the parser to start constructing an A-chain. A *trace* of the DP is placed in the A-movement stack. Since there is now a trace awaiting attachment, a position satisfying its Case and θ requirements will be searched for in subsequent processing.

Store	Contents
Input	the cat
Buffer	$[_{\text{IP}}\ [_{\text{DP}}$ the dog$]_i\ [_{\text{I}'}\ [_{\text{I}}$ +past]]] – $[_{\text{VP}}\ [_{\text{V}'}\ [_{\text{V}}$ chase]]]
Stack	
A mvnt stack	t_i

There are currently at least two chunks in the buffer (in fact, exactly two), so the chunker is not called upon to replenish it. No attachments can be done since the stack is empty, and no chain construction can be pursued in this particular configuration either. Hence, the first element is removed from the buffer and pushed onto the stack.

Store	Contents
Input	the cat
Buffer	$[_{\mathrm{VP}}\ [_{\mathrm{V'}}\ [_{\mathrm{V}}\ \text{chase}]]]$
Stack	$[_{\mathrm{IP}}\ [_{\mathrm{DP}}\ \text{the dog}]_i\ [_{\mathrm{I'}}\ [_{\mathrm{I}}\ +\text{past}]]]$
A mvnt stack	t_i

The buffer now contains less than two chunks, and so the chunker becomes active in order to place more items in the buffer, thereby re-establishing the parser's lookahead. The chunker examines the input stream, and places the chunk *the cat* in the buffer.

Store	Contents
Input	
Buffer	$[_{\mathrm{VP}}\ [_{\mathrm{V'}}\ [_{\mathrm{V}}\ \text{chase}]]]$ – $[_{\mathrm{DP}}\ \text{the cat}]$
Stack	$[_{\mathrm{IP}}\ [_{\mathrm{DP}}\ \text{the dog}]_i\ [_{\mathrm{I'}}\ [_{\mathrm{I}}\ +\text{past}]]]$
A mvnt stack	t_i

The attacher finally gets its turn again. The VP and the IP cannot be attached to each other in any fashion,[8] so the movement stacks are checked. The A-movement stack contains a trace which requires a θ-role (but must not be assigned Case, as it has already been assigned Case marking), and the verb *chase* assigns a θ-role to the specifier position of its VP (but no Case marking). The attacher can therefore attach the trace into this position. As this A-chain is now complete, no further trace is placed into the A-movement stack.

---

[8] Recall that any phrase must be complete before it can be attached to another phrase; all (functional or lexical) selection must be satisfied first. Once a piece of phrase structure has been incorporated into another, this incorporated piece is unavailable for further manipulation; hence a correct parse could not possibly be achieved if an incomplete phrase were attached to some parse tree fragment.

Store	Contents
Input	
Buffer	$[_{\mathrm{VP}}\ t_i\ [_{\mathrm{V'}}\ [_{\mathrm{V}}\ \text{chase}]]] - [_{\mathrm{DP}}\ \text{the cat}]$
Stack	$[_{\mathrm{IP}}\ [_{\mathrm{DP}}\ \text{the dog}]_i\ [_{\mathrm{I'}}\ [_{\mathrm{I}}\ +\text{past}]]]$
A mvnt stack	

As nothing more can be done in this configuration, the first element of the buffer is pushed onto the stack.

Store	Contents
Input	
Buffer	$[_{\mathrm{DP}}\ \text{the cat}]$
Stack	$[_{\mathrm{VP}}\ t_i\ [_{\mathrm{V'}}\ [_{\mathrm{V}}\ \text{chase}]]] - [_{\mathrm{IP}}\ [_{\mathrm{DP}}\ \text{the dog}]_i\ [_{\mathrm{I'}}\ [_{\mathrm{I}}\ +\text{past}]]]$
A mvnt stack	

The buffer once again contains fewer than two chunks, but since the input stream is empty the chunker is unable to replenish the buffer. Processing thus continues in this configuration. The DP *the cat* requires both Case marking and a θ-role, and *chase* assigns Case and a θ-role to the complement of the VP position; the attacher carries out the attachment. Since the Case and θ requirements of this DP are satisfied there is no A-movement going on, and no trace needs to be placed in the A-movement stack.

Store	Contents
Input	
Buffer	$[_{\mathrm{VP}}\ t_i\ [_{\mathrm{V'}}\ [_{\mathrm{V}}\ \text{chase}]\ [_{\mathrm{DP}}\ \text{the cat}]]]$
Stack	$[_{\mathrm{IP}}\ [_{\mathrm{DP}}\ \text{the dog}]_i\ [_{\mathrm{I'}}\ [_{\mathrm{I}}\ +\text{past}]]]$
A mvnt stack	

The VP is finally complete, and so it and the IP can be connected. This forms the final parse tree.

Store	Contents
Input	
Buffer	$[_{\text{IP}}$ $[_{\text{DP}}$ the dog$]_i$ $[_{\text{I}'}$ $[_{\text{I}}$ +past] $[_{\text{VP}}$ t_i $[_{\text{V}'}$ $[_{\text{V}}$ chase] $[_{\text{DP}}$ the cat]]]]]
Stack	
A mvnt stack	

This structure is pushed onto the (empty) stack, to make sure that the buffer is also empty. At this point everything except the stack is empty. The stack contains a single element. The parse was thus successful.

Other A-type movements are handled in an analogous fashion.

4.4 *Wh*-movement

Wh-movement is a type of $\bar{\text{A}}$-movement. $\bar{\text{A}}$ dependencies are more difficult to determine than A dependencies. A dependencies are constrained by Case and θ theory; each member of an A chain is therefore very easy to identify. This is not the case for $\bar{\text{A}}$ chains, and the extremities of an $\bar{\text{A}}$ chain cause special problems for the parser (the foot causes difficulties in languages like English, the head in those like Chinese).

4.4.1 S-Structure *Wh*-movement

Let us now consider a case of S-Structure *wh*-movement, as is exhibited in,

(67) Who did you kiss?

The structure which the parser is to recover for this sentence is,

(68) $[_{\text{CP}}$ who$_i$ $[_{\text{IP}}$ did you $[_{\text{VP}}$ kiss t_i]]]

Again, the parser begins parsing in the following configuration.

Store	Contents
Input	who did you kiss
Buffer	
Stack	
X^0 mvnt stack	
A mvnt stack	
$\overline{A}$ mvnt stack	

The chunker reads items from the input stream, projecting a skeletal $\overline{X}$ structure for each, until there are at least two chunks in the buffer. It begins with the *wh* word *who*.[9]

Store	Contents
Input	did you kiss
Buffer	$[_{DP}$ who $]$
Stack	
X^0 mvnt stack	
A mvnt stack	
$\overline{A}$ mvnt stack	

Next the chunker encounters *did*. This word phonologically/morphologically reflects the presence of tense features. The syntactic theory holds that tense features are generated under the I node, and move, via X^0-movement, to the C node. The parser detects this X^0-movement, placing the appropriate tense features on the X^0-movement stack.

---

[9] In English it is clear that a *wh*-phrase situated at the front of a clause has been displaced (unless one adopts the Vacuous Movement Hypothesis [Chomsky 86a]), and one might question why the parser does not immediately begin constructing an $\overline{A}$-chain. While this reasoning applies to English, the parser does not know which language it is dealing with and thus cannot make any assumptions about the location of any element until it is attached into a a phrase structure fragment by the attacher. (The chunker does violate this principle in the case of head movement, as discussed in section 4.2.)

Store	Contents
Input	you kiss
Buffer	[$_{DP}$ who] – [$_{CP}$ [$_{C'}$ [$_{C}$ do]]]
Stack	
X^0 mvnt stack	[+past]
A mvnt stack	
$\overline{\text{A}}$ mvnt stack	

Nothing can be done in this configuration, so the first element in the buffer is removed and pushed onto the stack.

Store	Contents
Input	you kiss
Buffer	[$_{CP}$ [$_{C'}$ [$_{C}$ do]]]
Stack	[$_{DP}$ who]
X^0 mvnt stack	[+past]
A mvnt stack	
$\overline{\text{A}}$ mvnt stack	

The chunker is again required to “fill up” the buffer, so that it contains at least two chunks.

Store	Contents
Input	kiss
Buffer	[$_{CP}$ [$_{C'}$ [$_{C}$ do]]] – [$_{DP}$ you]
Stack	[$_{DP}$ who]
X^0 mvnt stack	[+past]
A mvnt stack	
$\overline{\text{A}}$ mvnt stack	

It is now possible for the attacher to attach the DP *who* to the specifier of CP position. Since this DP has not yet received Case marking nor a θ-role, it is clear that it must move. This movement must be of the $\overline{\text{A}}$ type; an A-chain is always headed by a Case marked position. A trace of the DP is thus placed on the $\overline{\text{A}}$ movement stack.

Store	Contents
Input	kiss
Buffer	$[_{\text{CP}}\ [_{\text{DP}}\ \text{who}\ ]_i\ [_{\text{C}'}\ [_{\text{C}}\ \text{do}\ ]]] - [_{\text{DP}}\ \text{you}\ ]$
Stack	
X^0 mvnt stack	[+past]
A mvnt stack	
$\overline{\text{A}}$ mvnt stack	t_i

Again, the first element from the buffer is removed and pushed onto the stack.

Store	Contents
Input	kiss
Buffer	$[_{\text{DP}}\ \text{you}\ ]$
Stack	$[_{\text{CP}}\ [_{\text{DP}}\ \text{who}\ ]_i\ [_{\text{C}'}\ [_{\text{C}}\ \text{do}\ ]]]$
X^0 mvnt stack	[+past]
A mvnt stack	
$\overline{\text{A}}$ mvnt stack	t_i

The next item from the input stream is analyzed by the chunker. This item is the verb *kiss*. The chunker knows that there must be an IP projection before the VP (through both functional selection and the presence of inflection on the verb), so an empty IP is placed in the buffer before the VP.

Store	Contents
Input	
Buffer	$[_{\mathrm{DP}}$ you $]$ – $[_{\mathrm{IP}}\ [_{\mathrm{I'}}\ [_{\mathrm{I}}\ ]]]$ – $[_{\mathrm{VP}}\ [_{\mathrm{V'}}\ [_{\mathrm{V}}$ kiss $]]]$
Stack	$[_{\mathrm{CP}}\ [_{\mathrm{DP}}$ who $]_i\ [_{\mathrm{C'}}\ [_{\mathrm{C}}$ do $]]]$
X^0 mvnt stack	[+past]
A mvnt stack	
$\bar{A}$ mvnt stack	t_i

Since nothing can be done in this configuration, the first element from the buffer is removed and pushed onto the stack.

Store	Contents
Input	
Buffer	$[_{\mathrm{IP}}\ [_{\mathrm{I'}}\ [_{\mathrm{I}}\ ]]]$ – $[_{\mathrm{VP}}\ [_{\mathrm{V'}}\ [_{\mathrm{V}}$ kiss $]]]$
Stack	$[_{\mathrm{DP}}$ you $]$ – $[_{\mathrm{CP}}\ [_{\mathrm{DP}}$ who $]_i\ [_{\mathrm{C'}}\ [_{\mathrm{C}}$ do $]]]$
X^0 mvnt stack	[+past]
A mvnt stack	
$\bar{A}$ mvnt stack	t_i

The empty IP is now at the front of the buffer, and the X^0 movement can be resolved. The tense features are deposited in the I node, and the X^0-movement stack is cleared of its contents.

Store	Contents
Input	
Buffer	$[_{\mathrm{IP}}\ [_{\mathrm{I'}}\ [_{\mathrm{I}}\ \text{+past}\ ]]] - [_{\mathrm{VP}}\ [_{\mathrm{V'}}\ [_{\mathrm{V}}\ \text{kiss}\ ]]]$
Stack	$[_{\mathrm{DP}}\ \text{you}\ ] - [_{\mathrm{CP}}\ [_{\mathrm{DP}}\ \text{who}\ ]_i\ [_{\mathrm{C'}}\ [_{\mathrm{C}}\ \text{do}\ ]]]$
X^0 mvnt stack	
A mvnt stack	
$\overline{\text{A}}$ mvnt stack	t_i

Since the I node contains tense features, it is now able to assign Case marking to the specifier of the IP node. This allows the attacher to attach the DP *you* into this position. This DP now has Case marking, but no θ-role, so an A-chain must be constructed. A trace of this DP, properly coindexed with an index unique from that of the $\overline{\text{A}}$ dependency also in progress, is placed in the A-movement stack.

Store	Contents
Input	
Buffer	$[_{\mathrm{IP}}\ [_{\mathrm{DP}}\ \text{you}\ ]_j\ [_{\mathrm{I'}}\ [_{\mathrm{I}}\ \text{+past}\ ]]] - [_{\mathrm{VP}}\ [_{\mathrm{V'}}\ [_{\mathrm{V}}\ \text{kiss}\ ]]]$
Stack	$[_{\mathrm{CP}}\ [_{\mathrm{DP}}\ \text{who}\ ]_i\ [_{\mathrm{C'}}\ [_{\mathrm{C}}\ \text{do}\ ]]]$
X^0 mvnt stack	
A mvnt stack	t_j
$\overline{\text{A}}$ mvnt stack	t_i

The first element of the buffer is removed and pushed onto the stack.

Store	Contents
Input	
Buffer	$[_{\mathrm{VP}}\ [_{\mathrm{V'}}\ [_{\mathrm{V}}$ kiss]]]
Stack	$[_{\mathrm{IP}}\ [_{\mathrm{DP}}$ you $]_j\ [_{\mathrm{I'}}\ [_{\mathrm{I}}$ +past]]] – $[_{\mathrm{CP}}\ [_{\mathrm{DP}}$ who $]_i\ [_{\mathrm{C'}}\ [_{\mathrm{C}}$ do]]]
X^0 mvnt stack	
A mvnt stack	t_j
$\overline{\mathrm{A}}$ mvnt stack	t_i

The VP is now at the front of the buffer. Since the specifier of VP position is assigned a θ-role but no Case marking, the A dependency can now be resolved. The trace is taken from the A movement stack, and is placed in the VP-internal subject position. Since all the Case and θ requirements of this A chain are now satisfied, no further trace is placed in the A-movement stack.

Store	Contents
Input	
Buffer	$[_{\mathrm{VP}}\ t_j\ [_{\mathrm{V'}}\ [_{\mathrm{V}}$ kiss]]]
Stack	$[_{\mathrm{IP}}\ [_{\mathrm{DP}}$ you $]_j\ [_{\mathrm{I'}}\ [_{\mathrm{I}}$ +past]]] – $[_{\mathrm{CP}}\ [_{\mathrm{DP}}$ who $]_i\ [_{\mathrm{C'}}\ [_{\mathrm{C}}$ do]]]
X^0 mvnt stack	
A mvnt stack	
$\overline{\mathrm{A}}$ mvnt stack	t_i

Although there are now less than two items in the buffer, the chunker cannot supply more chunks as the input stream is empty. There is nothing that the attacher can do in this configuration, so the VP is removed from the buffer and pushed onto the stack.

Store	Contents
Input	
Buffer	
Stack	$[_{\text{VP}}\ t_j\ [_{\text{V}'}\ [_{\text{V}}\ \text{kiss}\ ]]]$ – $[_{\text{IP}}\ [_{\text{DP}}\ \text{you}\ ]_j\ [_{\text{I}'}\ [_{\text{I}}\ \text{+past}\ ]]]$ – $[_{\text{CP}}\ [_{\text{DP}}\ \text{who}\ ]_i\ [_{\text{C}'}\ [_{\text{C}}\ \text{do}\ ]]]$
X^0 mvnt stack	
A mvnt stack	
$\overline{\text{A}}$ mvnt stack	t_i

The attacher is now considering the right edge of the VP; there is a Case- and θ-marked position to the right of the verb. Since there is no overt NP which can be placed in this position, the attacher consults the movement stacks to see if there is any trace which would be compatible with this position. Of course, the $\overline{\text{A}}$-chain can now be resolved.

Store	Contents
Input	
Buffer	$[_{\text{VP}}\ t_j\ [_{\text{V}'}\ [_{\text{V}}\ \text{kiss}\ t_i\ ]]]$
Stack	$[_{\text{IP}}\ [_{\text{DP}}\ \text{you}\ ]_j\ [_{\text{I}'}\ [_{\text{I}}\ \text{+past}\ ]]]$ – $[_{\text{CP}}\ [_{\text{DP}}\ \text{who}\ ]_i\ [_{\text{C}'}\ [_{\text{C}}\ \text{do}\ ]]]$
X^0 mvnt stack	
A mvnt stack	
$\overline{\text{A}}$ mvnt stack	

The VP, now complete, can be attached into the complement position of the IP. Functional selection licenses this attachment.

Store	Contents
Input	
Buffer	$[_{\text{IP}}$ $[_{\text{DP}}$ you $]_j$ $[_{\text{I}'}$ $[_{\text{I}}$ +past] $[_{\text{VP}}$ t_j $[_{\text{V}'}$ $[_{\text{V}}$ kiss t_i]]]]]
Stack	$[_{\text{CP}}$ $[_{\text{DP}}$ who $]_i$ $[_{\text{C}'}$ $[_{\text{C}}$ do]]]
X^0 mvnt stack	
A mvnt stack	
$\overline{\text{A}}$ mvnt stack	

Similarly, functional selection licenses the attachment of the complete IP into the complement of CP position. This results in a complete and correct parse tree for this sentence.

Store	Contents
Input	
Buffer	$[_{\text{CP}}$ $[_{\text{DP}}$ who $]_i$ $[_{\text{C}'}$ $[_{\text{C}}$ do] $[_{\text{IP}}$ $[_{\text{DP}}$ you $]_j$ $[_{\text{I}'}$ $[_{\text{I}}$ +past] $[_{\text{VP}}$ t_j $[_{\text{V}'}$ $[_{\text{V}}$ kiss t_i]]]]]]]
Stack	
X^0 mvnt stack	
A mvnt stack	
$\overline{\text{A}}$ mvnt stack	

4.4.2 LF *Wh*-movement

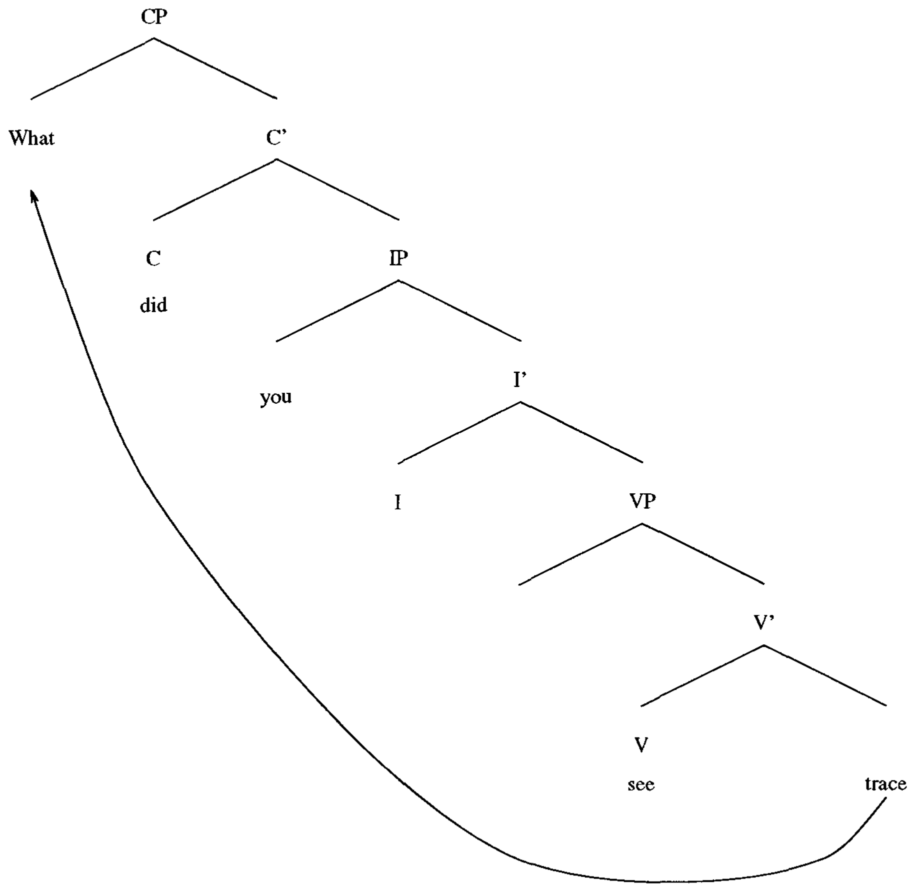
Given what has been presented so far, it does not appear that $\overline{\text{A}}$-movement should cause any fundamental difficulties for the parsing mechanism. However, if the parser is to operate in accordance with the minimal assumptions outlined, then LF *wh*-movement should be an unparsable construction. Languages such as Chinese have LF *wh*-movement, and it is to Chinese that we now turn our attention.

Some examples of direct questions in Chinese are shown below. "Shenme", the question word corresponding to the English "what", appears in its D-Structure position in all the sentences.

(69) a. Ni kanjian le shenme?
you see what
"What did you see?"
b. Ni shuo ni kanjian le shenme?
you say you see ASP what
"What did you say you saw?"
c. Ni renwei ni shuo ni kanjian le shenme?
you think you say you see ASP what
"What did you think you said you saw?"

It is clear that the question word can be arbitrarily far away from the position where it should appear at LF, the matrix clause specifier of CP (compare the English translations, where "what" occupies this position, and see figure 4.17). The difficulty which arises while the parser is processing the sentence is that it has no way of knowing that it is dealing with a *wh* construction until encounters the very last word of the sentence. It seems inevitable, given the standard analysis of *wh* movement, that the parser will have to use either unbounded lookahead or backtracking in order to produce an accurate parse tree.

It seems to be a fact of language that overt movement in general, and overt wh-movement in particular, is leftward. Rightward movement, while not ruled out, is severely constrained [Ross 67, Grosu 73]. While there is nothing within the theory which forces the movement to be leftward — no directionality of movement is imposed directly on the rule move-α — GB theory seems to make the strong assumption that all long distance movement, whether it be overt or covert, is leftward (except under those very constrained conditions where rightward movement can occur). From the left-to-right nature of the



(a) English

Figure 4.17: English LF representation

parser's operation and its filler-driven strategy for constructing movement chains, it falls out that the filler must come before (to the left of) the gap. A parsing-based explanation for this directional bias of move-α thus emerges.

At first sight, this explanation does not seem to help in the case of covert *wh*-movement; here the filler is located in its D-Structure position, and it seems that there is no way to relate such a *wh*-phrase to its proper LF position in the specifier of CP position. There is, however, a straightforward way to get around this difficulty — assume that covert movement is *rightward.* There is nothing in the syntactic theory to rule out the LF representation shown in figure 4.18.

With this LF analysis for Chinese *wh*-movement, the parser can construct chains in basically the same manner as for English. The parser does not start to construct a chain until it encounters the filler (the *wh* word). Instead of leaving the filler in this position and placing a trace in the $\overline{\text{A}}$-movement stack, the parser places the filler in the $\overline{\text{A}}$-movement stack, and leaves a properly coindexed trace behind. This filler is carried along towards the right until a final landing site for it is found; the parser will deposit intermediate traces along the way as required.

Two distinct types of filler-driven parsing are thus engaged in by the parser (as reported in [Alphonce *et al.* 92, Davis *et al.* 92]). The first is filler-driven *gap-locating* parsing (exemplified by English *wh*-movement), while the second is filler-driven *gap-creating* parsing (Chinese *wh*-movement). Gap-creating parsing allows the parser to start building a chain of covert movement when it has identified the filler. The filler is moved to the right, and a gap is created in the S-Structure position of the filler. Note that this is in contrast to the case of gap-locating movement, which leaves the filler in place and creates a trace which is then moved along.

It is easy for the parser to identify whether an element should undergo gap-locating or gap-creating movement as Case and θ conditions determine the type of movement that

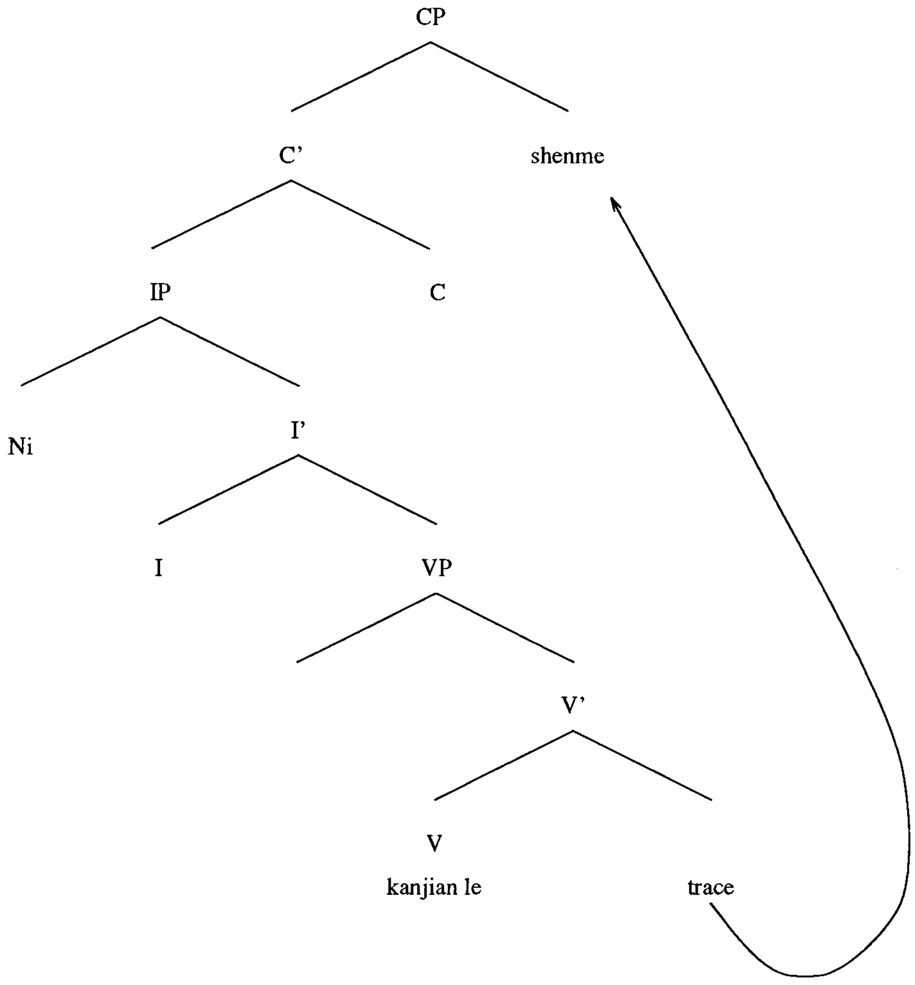


Figure 4.18: Proposed Chinese LF representation

is required (see table 4.2). The Case and θ properties of an element in a given position unambiguously identify the type of movement it must undergo in order to saturate its Case-marking and θ-role requirements. Consider an element which has Case-marking but no θ-role. Since the head of an A-chain is Case-marked while its foot is assigned a θ-role, clearly this element must undergo A-movement. Furthermore, since it is lacking a θ-role, it must be a case of gap-locating movement. In contrast, an element which has a θ-role but requires Case-marking must undergo gap-creating A-movement.

Filler-driven gap-locating parsing is used to recover S-Structure dependencies, while

	Type of Chain Formation			
	Gap Locating		Gap Creating	
	$\bar{\text{A}}$-movement	A-movement	A-movement	$\bar{\text{A}}$-movement
Case	needs	has	needs	has
θ-role	needs	needs	has	has

Table 4.2: Movement-type diagnostics

filler-driven gap-creating parsing is used to recover LF dependencies. Since movement chains can interfere with each other only if they are established at the same level of representation, the parser uses two sets of movement stacks, a gap-locating set and a gap-creating set.

Let us now consider an example of how the parser deals with Chinese *wh* movement. The sentence to be parsed is,

(70) Ni kanjian le shenme?

The assumed structure which the parser should recover is,

(71) $[_{\text{CP}}$ $[_{\text{IP}}$ Ni $[_{\text{VP}}$ kanjian le t_i]] shenme$_i$]

Note that "le" is an aspectual marker, and is treated by the parser as part of the verb.

The initial state of the parser is as in the previous cases,

Store	Contents
Input	ni kanjian-le shenme
Buffer	
Stack	

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\overline{\text{A}}$		

The chunker starts things off by processing the first item from the input stream. This first item is the pronoun "ni"; the chunker puts a DP chunk containing "ni" into the buffer.

Store	Contents
Input	kanjian-le shenme
Buffer	[$_{\text{DP}}$ ni]
Stack	

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\overline{\text{A}}$		

Since the buffer only contains one item, the chunker is called upon to provide more chunks. In processing the next word from the input, the verb "kanjian-le", two chunks are produced and placed in the buffer.

Store	Contents
Input	shenme
Buffer	[$_{\text{DP}}$ ni] – [$_{\text{IP}}$ [$_{\text{I}'}$ [$_{\text{I}}$ +asp]]] – [$_{\text{VP}}$ [$_{\text{V}'}$ [$_{\text{V}}$ kanjian-le]]]
Stack	

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\overline{\text{A}}$		

The first element from the buffer is pushed onto the stack.

Store	Contents
Input	shenme
Buffer	[$_{\text{IP}}$ [$_{\text{I}'}$ [$_{\text{I}}$ +asp]]] – [$_{\text{VP}}$ [$_{\text{V}'}$ [$_{\text{V}}$ kanjian-le]]]
Stack	[$_{\text{DP}}$ ni]

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\overline{\text{A}}$		

The DP is attached into the specifier of IP position. Since in Chinese verbs assign their external θ-roles to the specifier of IP position, the Case and θ properties of the DP are satisfied, and no A-chain is constructed (in contrast to the English case).

Store	Contents
Input	shenme
Buffer	[$_{IP}$ [$_{DP}$ ni] [$_{I'}$ [$_{I}$ +asp]]] – [$_{VP}$ [$_{V'}$ [$_{V}$ kanjian-le]]]
Stack	

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\bar{A}$		

Again, nothing can be done and the stack is empty; the first element of the buffer is therefore pushed onto the stack.

Store	Contents
Input	shenme
Buffer	[$_{VP}$ [$_{V'}$ [$_{V}$ kanjian-le]]]
Stack	[$_{IP}$ [$_{DP}$ ni] [$_{I'}$ [$_{I}$ +asp]]]

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\bar{A}$		

More chunks need to be placed in the buffer, as it now contains fewer than two. The chunker processes the next item from the input stream, “shenme”, and places the chunk based on it in the buffer.

Store	Contents
Input	
Buffer	$[_{\mathrm{VP}}\ [_{\mathrm{V}'}\ [_{\mathrm{V}}$ kanjian-le]]] – $[_{\mathrm{DP}}$ shenme]
Stack	$[_{\mathrm{IP}}\ [_{\mathrm{DP}}$ ni] $[_{\mathrm{I}'}\ [_{\mathrm{I}}$ +asp]]]

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\overline{\mathrm{A}}$		

The first element from the buffer is pushed onto the stack.

Store	Contents
Input	
Buffer	$[_{\mathrm{DP}}$ shenme]
Stack	$[_{\mathrm{VP}}\ [_{\mathrm{V}'}\ [_{\mathrm{V}}$ kanjian-le]]] – $[_{\mathrm{IP}}\ [_{\mathrm{DP}}$ ni] $[_{\mathrm{I}'}\ [_{\mathrm{I}}$ +asp]]]

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\overline{\mathrm{A}}$		

Since the input stream is empty, no more chunks can be added to the buffer. Hence processing must continue in the present configuration; a gap-creating movement is initiated. "Shenme" is a *wh*-phrase which needs to move at LF to the specifier of some CP. It is to be attached into a position which is both Case-marked and assigned a θ-role. This is the indication to the parser that it must construct a chain using gap-creation. Therefore, a trace is inserted into the complement position of the VP, and the DP itself is placed in the gap-creating $\overline{\mathrm{A}}$-movement stack.

Store	Contents
Input	
Buffer	[$_{\text{VP}}$ [$_{\text{V}'}$ [$_{\text{V}}$ kanjian-le] t_i]]
Stack	[$_{\text{IP}}$ [$_{\text{DP}}$ ni] [$_{\text{I}'}$ [$_{\text{I}}$ +asp]]]

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\overline{\text{A}}$		[$_{\text{DP}}$ shenme]$_i$

The VP is now complete, so its attachment into the IP, licensed by functional selection, is done.

Store	Contents
Input	
Buffer	[$_{\text{IP}}$ [$_{\text{DP}}$ ni] [$_{\text{I}'}$ [$_{\text{I}}$ +asp] [$_{\text{VP}}$ [$_{\text{V}'}$ [$_{\text{V}}$ kanjian-le] t_i]]]]
Stack	

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\overline{\text{A}}$		[$_{\text{DP}}$ shenme]$_i$

The parser can do nothing but push the first element from the buffer onto the stack in this configuration.

Store	Contents
Input	
Buffer	
Stack	$[_{\text{IP}}\ [_{\text{DP}}\ \text{ni}\ ]\ [_{\text{I}'}\ [_{\text{I}}\ \text{+asp}\ ]\ [_{\text{VP}}\ [_{\text{V}'}\ [_{\text{V}}\ \text{kanjian-le}\ ]\ t_i\ ]]]]$

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\bar{\text{A}}$		$[_{\text{DP}}\ \text{shenme}\ ]_i$

Although the stack contains a single phrase structure fragment and the buffer is empty, not all the movement stacks are void of content. Therefore, the parse is not yet complete. There is an $\bar{\text{A}}$-dependency which needs to be resolved. The presence of the *wh*-phrase in the gap-creating $\bar{\text{A}}$ movement stack, and the presence of a single IP in the stack, signals to the parser that the landing site for the *wh*-phrase must be created. The IP is dropped back into the buffer.

Store	Contents
Input	
Buffer	$[_{\text{IP}}\ [_{\text{DP}}\ \text{ni}\ ]\ [_{\text{I}'}\ [_{\text{I}}\ \text{+asp}\ ]\ [_{\text{VP}}\ [_{\text{V}'}\ [_{\text{V}}\ \text{kanjian-le}\ ]\ t_i\ ]]]]$
Stack	$[_{\text{CP}}\ [_{\text{C}'}\ [_{\text{C}}\ ]]]$

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\bar{\text{A}}$		$[_{\text{DP}}\ \text{shenme}\ ]_i$

The IP is attached into the complement position of the CP.

Store	Contents
Input	
Buffer	$[_{CP}\ [_{C'}\ [_{IP}\ [_{DP}$ ni $]\ [_{I'}\ [_{I}$ +asp $]\ [_{VP}\ [_{V'}\ [_{V}$ kanjian-le $]\ t_i\ ]]]]]]$
Stack	

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\bar{A}$		$[_{DP}$ shenme $]_i$

Nothing can be done, so the first element from the buffer is pushed onto the stack.

Store	Contents
Input	
Buffer	
Stack	$[_{CP}\ [_{C'}\ [_{IP}\ [_{DP}$ ni $]\ [_{I'}\ [_{I}$ +asp $]\ [_{VP}\ [_{V'}\ [_{V}$ kanjian-le $]\ t_i\ ]]]]]]$

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\bar{A}$		$[_{DP}$ shenme $]_i$

The $\bar{A}$-dependency can finally be resolved. The specifier of CP is on the right, and so the parser can in this configuration attach the DP from the movement stack into this position, completing the parse tree.

Store	Contents
Input	
Buffer	[$_{CP}$ [$_{C'}$ [$_{IP}$ [$_{DP}$ ni] [$_{I'}$ [$_{I}$ +asp] [$_{VP}$ [$_{V'}$ [$_{V}$ kanjian-le] t_i]]]]] [$_{DP}$ shenme]$_i$]
Stack	

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\bar{A}$		

Finally, the parser moves the completed parse tree into the stack, to ensure that the buffer is emtpy, which it is. The state is:

Store	Contents
Input	
Buffer	
Stack	[$_{CP}$ [$_{C'}$ [$_{IP}$ [$_{DP}$ ni] [$_{I'}$ [$_{I}$ +asp] [$_{VP}$ [$_{V'}$ [$_{V}$ kanjian-le] t_i]]]]] [$_{DP}$ shenme]$_i$]

	Gap Locating Movement Stack	Gap Creating Movement Stack
X^0		
A		
$\bar{A}$		

4.5 Revised Parsing Model

The parser engages in two types of chain-creation — gap-locating and gap-creating. Since these correspond to application of the rule move-α at different levels of the grammar, there will be no interference between chains constructed using gap-location and those built using gap-creation. Each type of chain-creating process can build any type of chain, either X^0, A, or $\overline{A}$. The set of movement stacks used to implement Relativized Minimality must therefore be refined to reflect the chain-creation process used. In other words, the parser has access to a set of gap-locating movement stacks and a set of gap-creating movement stacks.

4.6 Empty Operator Constructions

We will now study a type of construction which, again, poses a dilemma for the parser under the assumptions of sections 2.14 and 3.2. Since several detailed examples of how the parser functions have been presented earlier in the chapter, I will simply describe (without the aid of parser "snapshots") how the parser deals with these constructions. Example parses for these types of constructions are shown in appendix A.2.

Recall the examples of empty operator constructions:

(72) a. Jennifer is the woman to watch Marlene.
b. Jennifer is the woman to tell the police to watch Marlene.

(73) a. Jennifer is the woman to watch.
b. Jennifer is the woman to tell the police to watch.

Under the processing assumptions Browning's analysis (section 2.10.4) predicts that the parser must backtrack in order to parse either an SGC or an OGC. The structure which should follow "the woman" cannot be determined until the end of the clause. As the (b) sentences show, this may be an unbounded distance away. Given the parsing assumptions, processing difficulties seem inevitable. The path I choose to pursue is that of maintaining both the linguistic and computational assumptions made earlier, while challenging the structure assigned to these clauses.

Abandoning Browning's analysis, I must offer an alternative. Pesetsky [Pesetsky 82] suggests that PRO may appear on the right. If this proposal is extended to any non-Case marked specifier, a unified analysis emerges.

(74) a. $[_{\text{NP}}\ ]\ [_{\text{CP}} O_i\ [_{\text{IP}}\ [_{\text{I}'}\ [_{\text{VP}} \ldots]\ ]\ \text{pro}_i\ ]]$
b. $[_{\text{NP}}\ ]\ [_{\text{CP}}\ O_i\ [_{\text{IP}}\ [_{\text{I}'}\ [_{\text{VP}} \ldots t_i]\ ]\ \text{PRO}_{ARB}\ ]]$

The same structure is built at the start of the clause in both instances — an empty operator (which need not be assumed to be pro).

In the case of a subject-gap clause, the extended projection principle (EPP) requires the presence of a subject; pro is inserted *on the right*, yielding an $\overline{\text{A}}$-bound pro [Cinque 90]. It is coindexed with the operator, thus avoiding a violation of the prohibition against vacuous quantification.

In the case of an object-gap clause, the parser finds a gap in the V' node, and coindexes this gap with the operator. The EPP requires the presence of a subject, hence PRO is inserted. This PRO cannot be bound by the operator as well; it would have to be parasitic in some sense, but this would violate the anti c-command constraint on parasitic gaps.

A slight complication arises here. The specifier of VP position is a position in which a DP-trace can occur. The Empty Category Principle holds that such a trace must be properly governed, and thus governed. By the PRO theorem, PRO must be ungoverned. Therefore, PRO cannot remain in the specifier of VP position, but must instead move to the specifier of IP position (which is ungoverned in untensed clauses).

The parser constructs the proper structure for both subject-gap and object-gap clauses with no extra machinery. The difference between the two simply falls out as a result of the interaction of two linguistic principles, the EPP and the constraint against vacuous quantification.

Chapter 5

Implications

The purpose of this chapter is to discuss the various syntactic and computational implications of the parsing model I am proposing.

A basic assumption I make is that all parameterization to account for cross-linguistic variation must occur in the competence module; the performance module is not parameterized. Conversely, I also make the assumption that any competence constraint is parameterized, and varies cross-linguistically, while any constraint which applies without change to all languages is a performance constraint. This has far-reaching implications as to the division of labour between competence and performance.

A case in point is Relativized Minimality (RM). As presented by Rizzi [Rizzi 90], RM does not allow for any variation across languages. I am therefore forced to cast RM as an artifact of the processing mechanism rather than as a principle of grammar. The parsing model presented herein incorporates RM in a natural manner.

I adopt the hypothesis that all long-distance dependencies are parsed in a filler-driven manner. Together with the processing assumptions, this predicts that long-distance overt movement must always be leftward, and similarly that long-distance covert movement is rightward. Conversely, long-distance leftward movement must be overt, while long-distance rightward movement is forced to be covert.

The fact that the directionality of movement and the level of representation at which the movement takes place are directly linked yields a performance explanation for this curious linear asymmetry. Since this is a performance constraint, it is predicted that long-distance overt movement *in all languages* is leftward, and long-distance covert movement *in all languages* is rightward. As far as I know, this is the case. It is also predicted that the restrictiveness of overt rightward movement will be mirrored for covert leftward movement.

Moreover, it is predicted that all possible configurations for the CP projection should be instantiated. As reported in [Davis *et al.* 92, Alphonce *et al.* 92], all possibilities are actually present. Figure 5.19 illustrates the possibilities. Examples of languages associated with the various configurations are: (I) English; (II) Farsi, Vietnamese; (III) Vata; (IV) Chinese, Japanese.

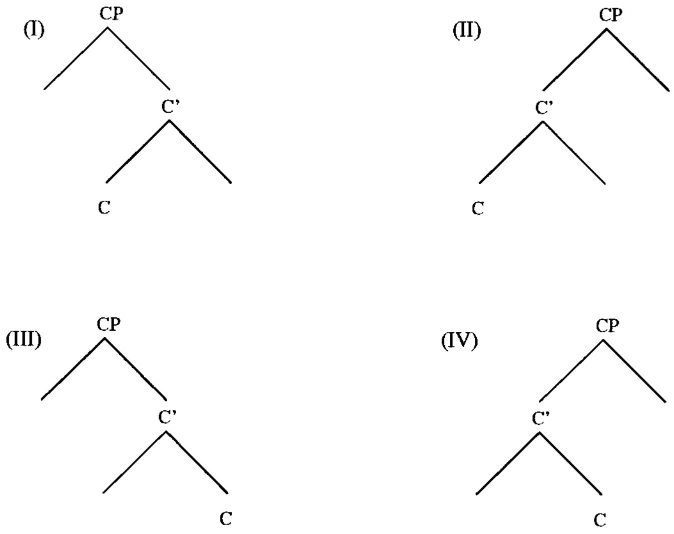


Figure 5.19: Possible CP projections

The fact that there is never an overt CP specifier in languages with LF *wh*-movement, another curiosity, now receives a straightforward explanation — languages with covert *wh*-movement do not have CP specifiers on the left.

The interaction of the linguistic with the processing assumptions forces abandonment of Browning's [Browning 87] proposed structure for infinitival relative clauses. Under Browning's analysis, one or the other of subject-gap and object-gap infinitival relative clauses should be unparsable, which clearly is not the case. I propose a structure which circumvents the processing difficulties, and which is in some ways a theoretically more pleasing analysis than that of Browning.

A fallout of the gap-locating and gap-creating mechanisms for chain construction is that it is entirely feasible to build a one pass deterministic parser based on a multistratal model of grammar. In particular, the multiple levels of GB theory pose no difficulty for a properly formulated parsing mechanism. Movement at different levels of the grammar are viewed by the parser simply as different types of chain building.

Furthermore, by adopting an appropriately articulated parsing model, it is reasonable to claim that this parsing scheme is applicable cross-linguistically. The practical consequences of this are numerous.

If one's goal is to produce multi-lingual applications, then an immediate benefit is that complex rule sets need not be developed for each new language. Once the parsing mechanism is in place, only the lexicon and the parameter settings need be specified (along with any language-specific information contained in the chunker, if any). A related benefit is that the amount of required maintenance for the parser is decreased. Testing and maintenance resources need not be distributed amongst a set of language-specific parser, but can all be concentrated on the single parsing mechanism.

The task of machine translation (at least as far as the syntactic component is concerned) is greatly simplified. Sharp [Sharp 85] developed an English-Spanish translation system based on GB theory, while Crocker [Crocker 88] developed an English-German one. Each system uses the same underlying mechanisms for both languages. Extending this idea, the implication is that a (relatively) simple translation method can be used for any pair of languages.

Chapter 6

Related Work

In this chapter I discuss some recent work which is related in some manner to the topic of this thesis. First I consider work directly concerned with the proper method to use in processing more than one language. Second I look at some other recent work which, while not directly concerned with cross-linguistic parsing at the moment, are nonetheless interesting.

6.1 Three Approaches

Mazuka & Lust [Mazuka *et al.* 90] argue that there are three different paths one can follow in parsing more than one language. One can, for example, claim that there is a single parsing scheme which is sufficient for all languages. That is the route that I have taken in this thesis, as have, for example, Frazier & Rayner [Frazier *et al.* 88] and Sato [Sato 88]. The other obvious road to take is to reject this hypothesis completely, and to build separate parsers for each new language. Lee, Chien, Lin, Huang, and Chen [Lee *et al.* 91] seem to take this route, at least with respect to English and Chinese. Finally, there is the middle ground, which Mazuka & Lust themselves occupy. They argue for a parsing scheme which is "universal" yet parameterized; by this they mean that there is a family of parsers for different languages, all of them deriving from the same basic components, and varying minimally due to a parameter setting. These proposals are considered briefly in the following three sections.

6.1.1 Language-Specific Parsing

Although their goal is not to produce a psycholinguistically plausible parsing scheme, Lee, Chien, Lin, Huang, and Chen [Lee *et al.* 91] are striving to produce an efficient parser for Chinese. They note that much work in natural language processing has been concerned with parsing "several western languages such as English" [p. 347]. Hypothesizing that a probable reason for the dearth of Chinese natural language processing systems stems from the fact that the syntactic structure of Chinese and English differ, they proceed to present a parser "specially designed for the Chinese language" [p. 347].

Although they refer to GB theory, their parser is based on a collection of phrase structure rules. They use a bidirectional chart parser, which could in itself be useful for a multitude of languages; the phrase structure rules are language-specific, however.

Thus, in tailoring their parser specifically to Chinese, they have limited the applicability of the parser to this one language. Although the bidirectionality of the parser renders it implausible as a model of human langauage processing, this feature of the parser could no doubt be exploited in parsing languages with radically different structures from that of Chinese.

6.1.2 Parameterized Parsing

Mazuka & Lust [Mazuka *et al.* 90] argue that certain constraints which Universal Grammar (UG) impose on possible grammars of natural languages directly influence the parsing mechanism used to process any given language. In particular, they hypothesize that there is a parameter in UG which determines the branching direction of a language, and that this parameter affects the form of the parsing mechanism. Thus, they reject the assumption I make that the parsing mechanism is fixed cross-linguistically.

They note that a top-down approach, commonly used for right-branching languages, predicts that left-branching constructions are difficult to parse. In conjunction with an assumption of universality of the parsing mechanism, this predicts that left-branching languages should be harder to parse than right-branching languages. This, however, is not the case. Mazuka & Lust therefore propose that right-branching languages, such as English, are parsed using a top-down approach, while left-branching languages, such as Japanese, use a bottom-up parsing algorithm. They claim that although universal, the human language processing system is parameterized; they claim further that this parameterization is due to the setting of a *competence* parameter, the *branching direction* parameter.

Mazuka & Lust themselves "recognize that neither a pure top-down nor a pure bottom-up approach will be appropriate for a complete model of natural language processing" [p. 181]. This is clearly the case for languages which exhibit mixed branching patterns, such as Chinese (which is head-final in its CP projection, but head-initial in its VP projection, for instance). A binary parameter based on branching direction seems unreasonable given the great diversity of branching patters cross-linguistically.

Hasegawa [Hasegawa 90] also casts doubt on their conclusion. The evidence which Mazuka & Lust cite does not unambiguously support their conclusion — they admit that "the existing data are not always consistent and in no way conclusive at this time" [p. 197]. Hasegawa further notes that "although it is an open question how grammar and parsing interact with each other ... it seems to be preferred, conceptually as well as methodologically, that these two components remain as separate modules, unless some strong evidence suggests otherwise." [p. 216] This is the null hypothesis to which I adhere.

6.1.3 Common Parsing Scheme

Sato [Sato 88] describes the Pattern Oriented Parser (POP), a bottom-up parser which relies on sentence and phrase patterns to parse both left-branching and right-branching languages.

POP builds parse trees according to sentence patterns. These patterns are "...parse tree frames, each of which is associated with one class of verbs..." [pg. 22] It is not clear whether chain wellformedness conditions are implemented, nor is it clear whether there is a fixed number of verb classes cross-linguistically. Even so, POP demonstrates that it certainly is possible to construct a single parsing mechanism which can handle both English and Japanese, in defiance of Mazuka & Lust's claim to the contrary.

Frazier & Rayner [Frazier *et al.* 88] argue for a universal parsing scheme which, due to the left or right branching nature of the language, takes on different forms to satisfy various processing constraints. Addressing the same problem as Mazuka & Lust they arrive at a fundamentally different solution. Frazier & Rayner are able to maintain the distinction between competence and performance which Mazuka & Lust abandon, as well as the notion of a completely universal parsing scheme. They propose a model which must satisfy constraints such as Late Closure and the First Analysis Constraint (discussed in section 3.3.2) among others. The idea is that the branching direction of the language, determined independently, causes these constraints to be satisfied in different ways by different languages, resulting in differing parsing strategies. As mentioned in section 4.1, this is an *indirect* means of explanation. I deny that the branching patterns of different languages force the use of a parsing mechanism with a variable amount of top-down versus bottom-up processing; I believe that a fundamentally bottom-up parser with certain top-down aspects is sufficient. Frazier & Rayner's approach does not, however, necessitate the abandonment of a strict division between competence and performance, nor does it invalidate the claim that the parser is not parameterized, as does Mazuka & Lust's proposal.

6.2 Other Research

6.2.1 Long-Distance $\overline{\text{A}}$-dependencies at S-Structure

Dahl, Popowich, and Rochemont [Dahl *et al.* 91] describe a GB based parsing system based on Static Discontinuity Grammars (SDGs). They translate GB theory into an SDG, a grammar formalism for which there are efficient parsing methods available. This research is not motivated by psycholinguistic concerns, but it does present some interesting ideas.

The parser does not explicitly construct multiple levels of representation, but constructs instead what they term D/S Structure. This is a representation at which the constraints of both D-Structure are S-Structure are enforced. My parser recovers an LF representation, with constraints at the various levels enforced at the appropriate time during tree construction. Since the grammar is embedded in the parser, this is possible. Dahl, Popowich, and Rochemont do not have this option available to them, as they encode the grammar using an SDG, which is then parsed using a generic SDG parsing algorithm.

6.2.2 A Grammar Tester

Fong [Fong 90] constructs a highly declarative parsing system which can be used as a grammar tester and a grammar-building tool by the the linguist. As such, if the grammar it is set up to implement can handle multiple languages, so can the parser. However, it is in no way psycholinguistically plausible, and therefore offers no insight into how the human language processing capacity might be structured.

Chapter 7

Future Work

There are many possible avenues to follow in future work. The first obvious path to travel is the cross-linguistic one: the parsing scheme should be tested on a greater variety of languages. Japanese is an interesting language, because it is a head-final language. Using the parser with Japanese would allow a direct test of Mazuka & Lust's [Mazuka *et al.* 90] claim that English and Japanese cannot be parsed using the same parsing scheme. Polish and Hungarian are interesting due to their representation of scope at S-Structure. There are also various *non-configurational* languages, among them Warlpiri, which do not have a fixed surface word order.

Successful incorporation of a greater number of movement processes, such as Quantifier Raising, and modules of the grammar, such as Binding, would further strengthen the claim that the general parsing scheme is viable.

A fairly simple extension to the parser, though one which is not implemented currently, involves having it handle binding chains in addition to antecedent government chains. Achieving this would involve making the $\overline{\mathrm{A}}$-movment stack somewhat more fine-grained. It would have to be split into an $\overline{\mathrm{A}}$/Binding movement stack as well as an $\overline{\mathrm{A}}$/Antecedent-Government movement stack. The linguistic implications of this would need to be investigated.

An analysis of various unmoved *wh*-phrases should also be undertaken. Unmoved *wh*-phrases include so-called *wh*-in-situ and "echo questions", respectively exemplified by,

(75) Who remembers what we bought where?

(76) Beverley saw Averill with *whom*?

The difficulty here is that the scope of the unmoved *wh*-phrase is obviously not established by overt *wh*-movement. Furthermore, it seems that it cannot be given by covert *wh*-movement either. The diffculty stems from the possible interpretations for these types of *wh*-phrases, which differs from that of usual *wh*-phrases.

An explanation for the difficulty in processing centre-embedding constructions, beyond the usual "memory-limitation" proposal, may be possible. Centre-embedding itself does not seem to lie at the root of people's processing difficulties, as cross-linguistically there are several centre-embedding structures which cause no such difficulties. In English *object relatives*, in which the DP being modified by the relative clause is interpreted with respect to the object position of the verb in the relative clause, are bad when nested. In contrast, while *subject relatives* (in which the modified DP is interpreted with respect to the subject position) can be nested without degradation.

(77) * The rat that the cat that the dog chased caught died.

(78) I saw the dog that chased the cat that caught the rat that died.

Interestingly, in Japanese and Chinese, nested object relatives are grammatical, while nested subject relatives are deviant. If one considers the movement of empty operators in these constructions, it seems as though improper interaction of chains of the same type is responsible for the processing difficulties, and not the fact that the constructions are centre-embedding. More study of this and other centre-embedding constructions is needed, however, before any firm conclusions can be drawn.

Chapter 8

Conclusion

This dissertation has investigated some issues which cast doubt on the existence of a universally applicable parsing scheme for natural language. Although I have not proved that there is a universal parsing scheme for all languages, it is hoped that the parsing model herein presented is a step in the proper direction.

This hope is based on the fact that the parser processes a right-branching language (English) in a primarily bottom-up manner (a mixed strategy, having both top-down and bottom-up aspects, is employed). Mazuka & Lust [Mazuka *et al.* 90] hold that there is a direct correspondence between the branching direction of a language and the top-down/bottom-up directionality of the parser; they claim that a top-down approach must be used for right-branching languages while left-branching languages require the use of a bottom-up approach. A mixed strategy should be sufficient to process a language of any branching pattern.

The parser also embodies mechanisms (gap-locating and gap-creating chain construction) which permit it to recover D-Structure and LF information from an S-Structure representation in a single-pass. This provides a solution to one of the problems encountered when basing a parser on a multistratal model of grammar.

Moreover, the parser adheres to (at least some of) the principles deemed necessary for a parser to be psycholinguistically plausible. Some aspects of its operation are, unfortunately, not plausible, yet these are related to the particular processing model on which I chose to base the parser; I do not believe that they will pose insurmountable difficulties.

Finally, in conformance with the hypothesis that all universal principles are performance principles and that all cross-linguistically varying conditions belong in the competence module, Relativized Minimality is cast as an artifact of processing constraints. This is one example of how processing considerations can yield explanations to linguistic phenomena where the syntactic theory could account for them only through stipulation. Other examples are the direction asymmetry exhibited by move-α, and the lack of overt CP specifiers on the left in languages with covert *wh*-movement.

Bibliography

[Abney 87] Steven Abney. Licensing and parsing. In Joyce McDonough and Bernadette Plunkett, editors, *Proceedings of NELS 17*, Amherst, Massachusetts, 1987. University of Massachusetts at Amherst.

[Abney 91] S. Abney. Parsing by chunks. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-based Parsing: Computation and Psycholinguistics.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.

[Abney *et al.* 86] Steven Abney and Jennifer Cole. A government-binding parser. In S. Berman, J.-W. Choe, and McDonough, editors, *Proceedings of NELS 16*, Amherst, Massachusetts, 1986. University of Massachusetts at Amherst.

[Alphonce *et al.* 92] Carl Alphonce and Henry Davis. Performance constraints and linguistic explanation. To appear in the proceedings of the Western Conference on Linguistics, 1992.

[Aoun *et al.* 87] J. Aoun, N. Hornstein, D. Lightfoot, and A. Weinberg. Two types of locality. *Linguistic Inquiry*, 18(4), 1987.

[Aoun *et al.* 89] J. Aoun and A. Li. Scope and constituency. *Linguistic Inquiry*, 20(2), 1989.

[Bresnan 82] J. Bresnan, editor. *The Mental Representation of Grammatical Relations.* MIT Press series on cognitive theory and mental representation. The MIT Press, 1982.

[Browning 87] M. Browning. *Null Operator Constructions.* PhD thesis, MIT, 1987.

[Chomsky 81] Noam Chomsky. *Lectures on Government and Binding.* Studies in Generative Grammar 9. Foris Publications, Dordrecht, The Netherlands, 1981.

[Chomsky 86a] Noam Chomsky. *Barriers.* Linguistic Inquiry Monographs 13. The MIT Press, 1986.

[Chomsky 86b] Noam Chomsky. *Knowledge of Language.* Convergence. Praeger Publishers, 1986.

[Chomsky 92] Noam Chomsky. A minimalist program for linguistic theory. Unpublished ms, MIT, 1992.

[Chomsky *et al.* 92] Noam Chomsky and Howard Lasnik. Principles and parameters theory. In J. Jacobs, A. von Stechow, and W. Sternefeld, editors, *Syntax: An International Handbook of Contemporary Research.* Walter de Gruyter, Berlin, 1992.

[Cinque 90] Guglielmo Cinque. *Types of* $\overline{A}$*- Dependencies.* Linguistic Inquiry Monographs 17. The MIT Press, 1990.

[Clocksin *et al.* 84] W. F. Clocksin and C. S. Mellish. *Programming in Prolog.* Springer-Verlag, second edition, 1984.

[Crocker 88] Matthew W. Crocker. A principle-based system for natural language analysis and translation. Master's thesis, Univeristy of British Columbia, 1988.

[Dahl *et al.* 91] Veronica Dahl, Fred Popowich, and Michael Rochemont. A principled characterization of dislocated phrases: Capturing barriers with static discontinuity grammars. Technical Report CMPT TR 91-09, School of Computing Science, Simon Fraser University, 1991.

[Davis 87] H. Davis. *The Acquisition of the English Auxiliary System and Its Relation to Linguistic Theory.* PhD thesis, University of British Columbia, 1987.

[Davis *et al.* 92] Henry Davis and Carl Alphonce. Parsing, WH-movement and linear asymmetry. To appear in the proceedings of the 22nd meeting of the North Eastern Linguistics Society, 1992.

[Fong 90] Sandiway Fong. *Computational Properties of Principle-Based Grammatical Theories.* PhD thesis, MIT Artificial Intelligence Laboratory, 1990.

[Frazier *et al.* 88] Lyn Frazier and Keith Rayner. Parameterizing the language processing system: Left- vs. right-branching within and across languages. In John A. Hawkins, editor, *Explaining Language Universals.* B. Blackwell, 1988.

[Frazier *et al.* 89] Lyn Frazier and Giovanni B. Flores D'Arcais. Filler driven parsing: A study of gap filling in Dutch. *Journal of Memory and Language*, 28(3), 1989.

[Gazdar *et al.* 85] Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar.* Basil Blackwell Publisher Ltd., Oxford, UK, 1985.

[Grosu 73]	A. Grosu. On the status of the so-called right roof constraint. *Language*, 49:294–311, 1973.
[Haegeman 91]	Liliane M. V. Haegeman. *Introduction to Government and Binding Theory.* B. Blackwell, 1991.
[Hasegawa 90]	Nobuko Hasegawa. Comments on Mazuka and Lust's paper. In L. Frazier and J. de Villiers, editors, *Language Processing and Language Acquisition*, pages 207–223. Kluwer, Dordrecht, 1990.
[Huang 82]	C.-T. J. Huang. *Logical Relations in Chinese and the Theory of Grammar.* PhD thesis, MIT, 1982.
[Johnsonbaugh 84]	Richard Johnsonbaugh. *Discrete Mathematics.* Macmillan Publishing Company, New York, NY, 1984.
[Kayne 84]	R. Kayne. *Connectedness and Binary Branching.* Foris, Dordrecht, 1984.
[Koopman *et al.* 91]	Hilda Koopman and Dominique Sportiche. The position of subjects. *Lingua*, 85:211–258, 1991.
[Kuroda 88]	S. Y. Kuroda. Whether we agree or not. In William J. Poser, editor, *Papers from the Second International Conference on Japanese Syntax.* Center for the Study of Language and Information, Stanford University, 1988.
[Kurtzman *et al.* 91]	Howard S. Kurtzman, Loren F. Crawford, and Caylee Nychis-Florence. Locating WH-traces. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-based Parsing: computation and psycholinguistics.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.
[Lasnik *et al.* 84]	H. Lasnik and M. Saito. On the nature of proper government. *Linguistic Inquiry*, 15:235–289, 1984.
[Lasnik *et al.* 88]	Howard Lasnik and Juan Uriagereka. *A Course in GB Syntax: Lectures on Binding and Empty Categories.* The MIT Press, 1988.
[LeBlanc 90]	David C. LeBlanc. The generation of phrase-structure representations from principles. Master's thesis, University of British Columbia, 1990.
[Lee *et al.* 91]	Lin-Shan Lee, Lee-Feng Chien, Long-Ji Lin, James Huang, and K.-J. Chen. An efficient natural language processing system specially designed for the Chinese language. *Computational Linguistics*, 17(4), 1991.

[Li 90]	Y. Audrey Li. *Order and Constituency in Mandarin Chinese.* Kluwer Academic Publishers, 1990.
[Mahajian 90]	A. Mahajian. *The A/A-bar Distinction and Movement Theory.* PhD thesis, MIT, 1990.
[Marcus 80]	Mitchell P. Marcus. *A Theory of Syntactic Recognition for Natural Language.* The MIT Press, 1980.
[Mazuka *et al.* 90]	R. Mazuka and B. Lust. On parameter-setting and parsing: Predictions for cross-linguistic differences in adult and child processing. In L. Frazier and J. de Villiers, editors, *Language Processing and Language Acquisition*, pages 163–205. Kluwer, Dordrecht, 1990.
[Pesetsky 82]	D. Pesetsky. *Paths and Categories.* PhD thesis, MIT, 1982.
[Pesetsky 87]	D. Pesetsky. Wh-in-situ: Movement and unselective binding. In E. Reuland and A. ter Meulen, editors, *The Representation of (In)definiteness.* The MIT Press, Cambridge, Massachusetts, 1987.
[Rensink *et al.* 91]	Ronald A. Rensink and Gregory Provan. The analysis of resource-limited vision systems. In *Program of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 311–316. Lawrence Erlbaum Associates, 1991.
[Rizzi 90]	L. Rizzi. *Relativized Minimality.* Linguistic Inquiry Monographs 16. The MIT Press, 1990.
[Ross 67]	J. R. Ross. *Constraints on variables in syntax.* PhD thesis, MIT, 1967. Distributed by the Indiana University Linguistics Club, Bloomington.
[Sato 88]	Paul T. Sato. A common parsing scheme for left- and right-branching languages. *Computational Linguistics*, 14(1):20–30, 1988.
[Sharp 85]	R. Sharp. A model of grammar based on principles of government and binding. Master's thesis, Univeristy of British Columbia, 1985.
[Speas 90]	M. Speas. *Phrase Structure in Natural Language.* Kluwer Academic Publishers, Dordrecht, 1990.
[van Riemsdijk *et al.* 86]	Henk van Riemsdijk and Edwin Williams. *Introduction to the Theory of Grammar.* The MIT Press, 1986.
[Williams 80]	Edwin Williams. Predication. *Linguistic Inquiry*, 11:203–238, 1980.

Appendix A

The Parser

A.1 Implementation

The parsing model described in the main body of the text has been implemented in Prolog, a logic programming language. The purpose of this appendix is to briefly describe the implementation, and provide several examples of the output of the parser.

The implementation faithfully models the parsing mechanism described in the text. The main predicate is `parse`, which embodies the attacher and the meta-level control in the parser. The input stream, the buffer, the main stack, and the various movement stacks are arguments of the predicate `parse`.

The chunker is implemented by a predicate named `chunk`. The input routine used by the chunker is taken from Clocksin & Mellish [Clocksin *et al.* 84].

Two versions of the parser have been compiled, one for Chinese and one for English. The only differences between the two versions are that,

1. the Chinese version consults a Chinese lexicon and subcategorization information, while the English version consults an English lexicon and subcategorization information, and
2. the parameter which indicates at which level *wh*-movement takes place is set to LF for Chinese and S-Structure for English, and
3. the branching directions for some $\overline{\mathrm{X}}$ projections differ, and
4. the chunking procedure differs slightly.

A.2 Examples

The purpose of this section is to present the parser's output for various examples. In order to make the Chinese examples easier to understand, a mini Chinese-English dictionary is provided below.

Nouns

Chinese	English
gou	dog
mao	cat
ni	you
shenme	what
shui	who

Verbs

Chinese	English
da	hit
mai	buy
xiangxin	believe
xiangzhidao	wonder
xihuan	like
zhui	chase

Miscellaneous

Chinese	English
bei	passive marker
le	aspectual marker

To begin with consider the English example shown below. It involves a simple A-dependency due to the movement of the subject from the VP-internal position into the IP specifier. Indices are printed in angled brackets, as in < gla = 1 >. Indices are identified according to the type of chain building procedure which created them, as well as the type of chain they are in. For example, *glab* signifies a *gap locating* $\overline{A}$-chain, while *gca* is the label associated with a *gap creating* *A*-chain. The indices for head movement chains (which are labelled *glh* or *glc*) are placed next to the maximal projection of the head, rather than next to the head itself.

```
[i2 < >
  [d2 < gla=1 >
    ec
    [d1
      d = the
      [n2 < >
        ec
        [n1
          n = dog
          ec]]]]
  [i1
    i = past
    [v2 < >
      [d2  trace  ] < gla=1 >
      [v1
        v = chase
        [d2 < >
          ec
          [d1
            d = the
            [n2 < >
              ec
              [n1
                n = cat
                ec]]]]]]]]

the dog chased the cat.
```

This next example is the same sentence, but this time in Chinese. It is a very simple example for the parser to deal with, since there is no movement going on.

```
[i2 < >
  [d2 < >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = gou
          ec]]]]]
  [i1
    i = aspect
    [v2 < >
      ec
      [v1
        v = zhui
        [d2 < >
          ec
          [d1
            d = ec
            [n2 < >
              ec
              [n1
                n = mao
                ec]]]]]]]]]

gou zhui mao.
```

This is an example of raising; note the A-chain. Since there is no Subject Raising in Chinese, there is no corresponding example in Chinese.

```
[i2 < >
  [d2 < gla=1 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = john
          ec]]]]
  [i1
    i = present
    [v2 < >
      [d2  trace  ] < gla=1 >
      [v1
        v = seem
        [i2 < >
          [d2  trace  ] < gla=1 >
          [i1
            i = to
            [v2 < >
              [d2  trace  ] < gla=1 >
              [v1
                v = like
                [d2 < >
                  ec
                  [d1
                    d = ec
                    [n2 < >
                      ec
                      [n1
                        n = mary
                        ec]]]]]]]]]]]]]

john seems to like mary.
```

Here we have passive sentence, first in English, then in Chinese. Recall that the parser treats the aspectual marker "le" as part of the verb (hence "`da_le`").

```
[i2 < >
  [d2 < gla=1 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = bill
          ec]]]]
  [i1
    i = past
    [v2 < >
      [d2  trace  ] < gla=1 >
      [v1
        v = be
        [v2 < >
          [d2  trace  ] < gla=1 >
          [v1
            v = hit
            [d2  trace  ] < gla=1 >]]]]]]

bill was hit.
```

```
[i2 < >
  [d2 < gla=1 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = bill
          ec]]]]
  [i1
    i = bei
    [v2 < >
      ec
      [v1
        v = da_le
        [d2  trace  ] < gla=1 >]]]]

bill bei da_le.
```

This is a simple $\overline{\text{A}}$-movement, a direct question. Compare the output for English and Chinese.

```
[c2 < glh=1 >
  [d2 < glab=2 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = who
          ec]]]]
  [c1
    c = do
    [i2 < glh=1 >
      [d2 < gla=3 >
        ec
        [d1
          d = ec
          [n2 < >
            ec
            [n1
              n = you
              ec]]]]
      [i1
        i = past
        [v2 < >
          [d2  trace  ] < gla=3 >
          [v1
            v = hit
            [d2  trace  ] < glab=2 >]]]]]]

who did you hit.
```

```
[c2 < >
  [c1
    [i2 < >
      [d2 < >
        ec
        [d1
          d = ec
          [n2 < >
            ec
            [n1
              n = ni
              ec]]]]
      [i1
        i = aspect
        [v2 < >
          ec
          [v1
            v = da_le
            [d2  trace  ] < gcab=1 >]]]]
    c = ec]
  [d2 < gcab=1 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = shui
          ec]]]]]

ni da_le shui.
```

This is a slightly more interesting case of $\overline{\text{A}}$-movement, an indirect question. Notice that in the Chinese case, the parser uses the verb's selectional restrictions (the verb "wonder" requires a [+*wh*] CP as complement) to decide where the $\overline{\text{A}}$-chain is to end.

```
[i2 < >
  [d2 < gla=1 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = john
          ec]]]]
  [i1
    i = past
    [v2 < >
      [d2  trace  ] < gla=1 >
      [v1
        v = wonder
        [c2 < >
          [d2 < glab=3 >
            ec
            [d1
              d = ec
              [n2 < >
                ec
                [n1
                  n = what
                  ec]]]]
          [c1
            c = ec
            [i2 < >
              [d2 < gla=2 >
                ec
                [d1
                  d = ec
                  [n2 < >
                    ec
                    [n1
                      n = bill
                      ec]]]]
              [i1
                i = past
                [v2 < >
                  [d2  trace  ] < gla=2 >
                  [v1
                    v = buy
                    [d2  trace  ] < glab=3 >]]]]]]]]]]

john wondered what bill bought.
```

```
 [i2 < >
   [d2 < >
     ec
     [d1
       d = ec
       [n2 < >
         ec
         [n1
           n = john
           ec]]]]
   [i1
     i = aspect
     [v2 < >
       ec
       [v1
         v = xiangzhidao
         [c2 < >
           [c1
             [i2 < >
               [d2 < >
                 ec
                 [d1
                   d = ec
                   [n2 < >
                     ec
                     [n1
                       n = bill
                       ec]]]]
               [i1
                 i = aspect
                 [v2 < >
                   ec
                   [v1
                     v = mai_le
                     [d2  trace  ] < gcab=1 >]]]]
             c = ec]
           [d2 < gcab=1 >
             ec
             [d1
               d = ec
               [n2 < >
                 ec
                 [n1
                   n = shenme
                   ec]]]]]]]]]

john xiangzhidao bill mai_le shenme.
```

This example is very similar to the previous one, except that here the matrix clause verb selects for an embedded clause which is [−*wh*].

```
[c2 < glh=1 >
  [d2 < glab=2 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = what
          ec]]]]
  [c1
    c = do
    [i2 < glh=1 >
      [d2 < gla=3 >
        ec
        [d1
          d = ec
          [n2 < >
            ec
            [n1
              n = john
              ec]]]]
      [i1
        i = past
        [v2 < >
          [d2  trace  ] < gla=3 >
          [v1
            v = believe
            [c2 < >
              [d2  trace  ] < glab=2 >
              [c1
                c = that
                [i2 < >
                  [d2 < gla=4 >
                    ec
                    [d1
                      d = ec
                      [n2 < >
                        ec
                        [n1
                          n = bill
                          ec]]]]
                  [i1
                    i = past
                    [v2 < >
                      [d2  trace  ] < gla=4 >
                      [v1
                        v = buy
                        [d2  trace  ] < glab=2 >]]]]]]]]]]]]
what did john believe bill bought.
```

```
[c2 < >
 [c1
  [i2 < >
   [d2 < >
    ec
    [d1
     d = ec
     [n2 < >
      ec
      [n1
       n = john
       ec]]]]
   [i1
    i = aspect
    [v2 < >
     ec
     [v1
      v = xiangxin
      [c2 < >
       [c1
        [i2 < >
         [d2 < >
          ec
          [d1
           d = ec
           [n2 < >
            ec
            [n1
             n = bill
             ec]]]]
         [i1
          i = aspect
          [v2 < >
           ec
           [v1
            v = mai_le
            [d2  trace  ] < gcab=1 >]]]]
        c = ec]
       [d2  trace  ] < gcab=1 >]]]]]
  c = ec]
 [d2 < gcab=1 >
  ec
  [d1
   d = ec
   [n2 < >
    ec
    [n1
     n = shenme
     ec]]]]]

john xiangxin bill mai_le shenme.
```

This is a case of topicalization, another $\overline{\mathrm{A}}$-movement. Notice that the movement is leftward in both English and Chinese.

```
[i: adjunct
  [d2 < glab=2 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = john
          ec]]]]
  [i2 < >
    [d2 < gla=1 >
      ec
      [d1
        d = ec
        [n2 < >
          ec
          [n1
            n = mary
            ec]]]]
    [i1
      i = present
      [v2 < >
        [d2  trace  ] < gla=1 >
        [v1
          v = like
          [d2  trace  ] < glab=2 >]]]]]

john mary likes.


[i: adjunct
  [d2 < glab=1 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = john
          ec]]]]
  [i2 < >
    [d2 < >
      ec
      [d1
        d = ec
        [n2 < >
          ec
          [n1
            n = mary
            ec]]]]
    [i1
      i = aspect
      [v2 < >
        ec
        [v1
          v = xihuan
          [d2  trace  ] < glab=1 >]]]]]

john mary xihuan.
```

A subject-gap infinitival relative clause in English.

```
 [i2 < >
   [d2 < gla=1 >
     ec
     [d1
       d = ec
       [n2 < >
         ec
         [n1
           n = john
           ec]]]]
   [i1
     i = past
     [v2 < >
       [d2  trace  ] < gla=1 >
       [v1
         v = kiss
         [d: adjunct
           [d2 < glab=2 >
             ec
             [d1
               d = the
               [n2 < >
                 ec
                 [n1
                   n = woman
                   ec]]]]
           [c2 < >
             [op2 < glab=2 >
               ec
               [op1
                 op = empty_op
                 ec]]
             [c1
               c = ec
               [i2 < >
                 [i1
                   i = to
                   [v2 < >
                     [v1
                       v = watch
                       [d2 < >
                         ec
                         [d1
                           d = ec
                           [n2 < >
                             ec
                             [n1
                               n = bill
                               ec]]]]]
                     [d2 < glab=2 >
                       ec
                       [d1
                         d = pro
                         ec]]]]
                ec]]]]]]]]
john kissed the woman to watch bill.
```

An object-gap infinitival relative clause in English.

```
[i2 < >
  [d2 < gla=1 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = john
          ec]]]]
  [i1
    i = past
    [v2 < >
      [d2  trace  ] < gla=1 >
      [v1
        v = kiss
        [d: adjunct
          [d2 < glab=2 >
            ec
            [d1
              d = the
              [n2 < >
                ec
                [n1
                  n = woman
                  ec]]]]
          [c2 < >
            [op2 < glab=2 >
              ec
              [op1
                op = empty_op
                ec]]
            [c1
              c = ec
              [i2 < >
                [i1
                  i = to
                  [v2 < >
                    [v1
                      v = watch
                      [op2  trace  ] < glab=2 >]
                    [d2  trace  ] < gca=3 >]]
                [d2 < gca=3 >
                  ec
                  [d1
                    d = big_PRO_arb
                    ec]]]]]]]]]]

john kissed the woman to watch.
```

The next example is an ambiguous sentence — is it [the man to kiss the woman] who John expected to watch the dog, or [the woman to watch the dog] who John expected the man to kiss? The parser only pursues a single parse, and comes up with the latter analysis.

```
[i2 < >
  [d2 < gla=1 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = john
          ec]]]]
  [i1
    i = past
    [v2 < >
      [d2  trace  ] < gla=1 >
      [v1
        v = expect
        [i2 < >
          [d2 < gla=2 >
            ec
            [d1
              d = the
              [n2 < >
                ec
                [n1
                  n = man
                  ec]]]]
          [i1
            i = to
            [v2 < >
              [d2  trace  ] < gla=2 >
              [v1
                v = kiss
                [d: adjunct
                  [d2 < glab=3 >
                    ec
                    [d1
                      d = the
                      [n2 < >
                        ec
                        [n1
                          n = woman
                          ec]]]]
                  [c2 < >
                    [op2 < glab=3 >
                      ec
                      [op1
                        op = empty_op
                        ec]]
                    [c1
                      c = ec
                      [i2 < >
                        [i1
                          i = to
                          [v2 < >
                            [v1
                              v = watch
                              [d2 < >
                                ec
                                [d1
                                  d = the
                                  [n2 < >
                                    ec
                                    [n1
                                      n = dog
                                      ec]]]]]
                            [d2 < glab=3 >
                              ec
                              [d1
                                d = pro
                                ec]]]]
                        ec]]]]]]]]]]]]

john expected the man to kiss the woman to watch the dog.
```

This is a case of an Exceptional Case Marking (ECM) verb in a context where the subject of the embedded clause receives its Case marking from the I node of the embedded clause. Thus, this is not an ECM structure.

```
[i2 < >
  [d2 < gla=1 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = mary
          ec]]]]
  [i1
    i = present
    [v2 < >
      [d2  trace  ] < gla=1 >
      [v1
        v = believe
        [i2 < >
          [d2 < gla=2 >
            ec
            [d1
              d = ec
              [n2 < >
                ec
                [n1
                  n = john
                  ec]]]]
          [i1
            i = present
            [v2 < >
              [d2  trace  ] < gla=2 >
              [v1
                v = be
                [d2 < >
                  ec
                  [d1
                    d = a
                    [n2 < >
                      ec
                      [n1
                        n = liar
                        ec]]]]]]]]]]]]
```

```
mary believes john is a liar.
```

Finally, a case of an (ECM) structure where the ECM verb assigns Case to the subject of the embedded clause.

```
[i2 < >
  [d2 < gla=1 >
    ec
    [d1
      d = ec
      [n2 < >
        ec
        [n1
          n = mary
          ec]]]]
  [i1
    i = present
    [v2 < >
      [d2  trace  ] < gla=1 >
      [v1
        v = believe
        [i2 < >
          [d2 < gla=2 >
            ec
            [d1
              d = ec
              [n2 < >
                ec
                [n1
                  n = john
                  ec]]]]
          [i1
            i = to
            [v2 < >
              [d2  trace  ] < gla=2 >
              [v1
                v = be
                [d2 < >
                  ec
                  [d1
                    d = a
                    [n2 < >
                      ec
                      [n1
                        n = liar
                        ec]]]]]]]]]]]]]

mary believes john to be a liar.
```