Theory and Parsing of the Coordinate Conjunction "And"

Victoria L. Snarr

Technical Report CSRI-171
September 1985

Computer Systems Research Institute
University of Toronto
Toronto, Canada
M5S 1A1

The Computer Systems Research Institute (CSRI) is an interdisciplinary group formed to conduct research and development relevant to computer systems and their application. It is jointly administered by the Department of Electrical Engineering and the Department of Computer Science of the University of Toronto, and is supported in part by the Natural Sciences and Engineering Research Council of Canada.

Theory and Parsing of the Coordinate Conjunction "And"

by

Victoria L. Snarr

Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
September 1984

A thesis submitted
in conformity with the requirements
for the degree of Master of Science

Abstract

Although the conjunction *and* appears to have a simple function in the English language, it has proved to be a stumbling block for both theoretical and computational linguists.

One of the theoretical problems of conjunction is to determine what governs the acceptability of a structure in which two elements are connected by *and*. The corresponding computational problem is, given this knowledge, to incorporate it into an efficient parser for English.

This thesis proposes a solution to the theoretical problem which is in the form of two general constraints — a syntactic constraint and a semantic one; and then incorporates these constraints into a "strictly deterministic" parser for English.

Acknowledgements

I would like to thank my thesis supervisor Ray Perrault for introducing me to computational linguistics and directing me to the topic of this thesis. ("If you're going to do a thesis on verb phrase ellipis, shouldn't you take a look at conjunction first?") I thank both Ray and my second thesis reader Graeme Hirst for their helpful comments and suggestions during the development of the thesis.

Special thanks go to Andrew Gullen who helped in so many ways — from reading and commenting on drafts of the thesis, to setting up "troff" macros to print the final version. I also want to thank all of my friends (whose names appear in the example sentences in the thesis) for making the last three years some of the most enjoyable I've had.

Special thanks also go to my family: my parents, Bill and Joyce, who have always been there when I needed them — providing both emotional and material support; and my sisters and brothers-in-law, Janet and Juan, Kathy and Brock, and Margie (Pidge). I also thank the Carter family for all they have done for me over the last few years.

Above all, I thank Chris who between cycling, windsurfing, kayaking, and running managed to give me all the love and encouragement one could ask for. This thesis is for him.

Contents

0. Introduction 1

Part I

1. Previous Proposals and Problems 3
2. A Theory of Coordinate Constituent Conjunction 7
   - 2.1 A Syntactic Constraint on CCC 7
   - 2.2 A Semantic Constraint on CCC 9
   - 2.3 A Necessary and Sufficient Condition for CCC 11
   - 2.4 A Model of Syntax 13
     - 2.4.1 VP Structure 13
     - 2.4.2 NP Structure 15
     - 2.4.3 Comp-S Structure 16
     - 2.4.4 Traces 17
   - 2.5 A Model of Predicate/Argument Relationships 18
     - 2.5.1 A Simple Case Grammar for English Verbs 18
     - 2.5.2 General Verb/Argument Relationships 23
     - 2.5.3 Preposition/Argument Relationships 25
     - 2.5.4 Dummy Predicates 26
     - 2.5.5 Conjunctions 28
   - 2.6 Summary 29

Part II

3. Parsing Coordinate Constituent Conjunctions 31
   - 3.1 The Parsing System 31
     - 3.1.1 The Parser 31
     - 3.1.2 The Predicate/Argument Analyser 38
     - 3.1.3 Interaction between the Parser and the Predicate/Argument Analyser 42
   - 3.2 A Method of Parsing Coordinate Constituent Conjunctions 43
     - 3.2.1 A First Attempt 43
     - 3.2.2 The Method 45
     - 3.2.3 The Grammar Rules for Parsing Conjunctions 51
     - 3.2.4 An Example 62
   - 3.3 Summary 67

4. Conclusions and Future Work 70

Appendix A. Grammar Rules for the Parser 75
Appendix B. Monitor Programs for the Predicate/Argument Analyser 85

References 86

0. Introduction

This thesis is concerned with the English coordinate conjunction *and*. Coordinate conjunctions, as the term implies, connect or "conjoin" elements of equal rank in sentences. A conjunct in a coordinate structure can be almost any grammatical constituent, or a constituent from which some elements have been elided. Five examples of coordinate conjunction are given below.

(1) Joe and Bill share an apartment.
(2) Karen went to the store and bought a radio.
(3) Jill solved a difficult and complex problem.
(4) Jack lives in Montreal and Peter, in Toronto.
(5) Susan owns, and Sharon wants to buy, an Italian sports car.

We will be concerned only with the coordinate conjunction of complete constituents like those in (1), (2), and (3) above, although many of the results can be generalized to handle conjunction of incomplete constituents like the ones in (4) and (5).

The thesis is divided into two parts. In **Part I**, we look at coordinate conjunction from a theoretical point of view and attempt to determine exactly what governs the acceptability of a coordinate structure in a sentence. (Our definition of acceptability excludes those coordinate structures which are peculiar for stylistic effect. In particular, the use of literary devices such as *syllepsis* and *zeugma* will be considered outside our domain.)

While it is clear that constituents which are coordinately conjoined must have something in common for the coordinate structure to be acceptable, the formalization of that commonality in an elegant fashion has proved to be a difficult problem for linguists. Our investigation of the problem led to the proposal of two general constraints — a syntactic constraint and a semantic one — which together provide a necessary and sufficient condition for the acceptability of the coordinate conjunction of two constituents in an English sentence.

In **Part II** of the thesis, the two constraints are incorporated into a parser which operates strictly deterministicly; that is, one which does not include a general backtracking mechanism. For the parser to be strictly deterministic was desirable from both a computational standpoint: the major source of inefficiency in many systems handling conjunction is the extensive backtracking that takes place during the parse of a coordinate structure; and from a linguistic standpoint: speakers of English appear to be able to parse most of the English sentences which occur naturally, even those containing conjunctions, without misanalysing substructures and having to backtrack.

PART I

1. Previous Proposals and Problems

It has been proposed by many linguists (e.g., Gazdar [1981]; Schacter [1977]) that for a coordinate structure to be acceptable, the two conjuncts must belong to the same syntactic category. Identity of syntactic category has also been proposed as a sufficient condition for coordinate conjunction (Gazdar [1981]).

Because of the interdependence between such constraints and the particular grammar assumed by the linguist, the constraints are reasonable only if the grammar rules that are derived from them account for the corresponding syntactic structures and do not interfere with rules accounting for the other syntactic structures of the language.

Gazdar [1981] proposes that the addition of rules derived from two phrase structure rule schemata are sufficient to account for all coordinate structures. These rules allow any two constituents belonging to the same major syntactic category (e.g., S, NP, VP) to be conjoined. However, given his grammar, this is not sufficiently restrictive. In fact, the problem of restricting the coordination of verbs was one of the motivations for the introduction of *subcategorizations* into the grammar (Gazdar [1982]). The following sentences are Gazdar's own examples.

(1) Kim threw and handed things to those outside.
(2) *Kim preferred and promised Sandy to go.

The grammaticality of (1) and the ungrammaticality of (2) are explained by the facts that the verbs *throw* and *hand* have the same verb subcategorization; and *prefer* and *promise* have different verb subcategorizations.

To constrain the constituents to have exactly the same subcategorizations is too strong; that is, to do so would block many grammatical conjunctions. For example, one of the subcategorization features in Gazdar's grammar is the feature [PAS] which is used in VP rules forming passive expressions. However passive and active VPs are often conjoined, as in the following sentence (again one of Gazdar's own examples).

(3) The Dodgers beat the Red Sox and were beaten by the Giants.

Therefore the set of subcategorization features relevant to coordinate conjunction must be determined. However, even if this can be achieved, there still remain two problems. The first is that it is not necessary for two constituents to belong to the same major syntactic category for their conjunction to be grammatical, as in sentence (4) below.[1]

---

1. Gazdar recognized the problem but, beyond the observation that "such expressions are permitted to appear in some slot in a sentence only when either conjunct could appear alone in that slot" (Gazdar [1981], p. 172), he had no explanation for such examples.

(4) He did the job slowly and with great care.

Schacter [1977] proposed a solution to this problem, which was to postulate a category such as Adverb Phrase, which dominates the entire coordinate construction and each of its immediate constituents. Although this works, the addition of new syntactic categories and rules to the grammar is not an appealing general solution as the inevitable result is a large grammar, which is undesirable from both computational and theoretical standpoints.

The second, more serious problem is that there are many unacceptable conjunctions (according to the definition given in the introduction) which cannot be accounted for at all using Gazdar's subcategorizations. For example, the grammar has no subcategorization feature which could be used to distinguish between grammatical sentences like sentence (4) and ungrammatical ones like sentence (5).

(4) He did the job slowly and with great care.
(5) *He did the job slowly and with a Swiss army knife.

Because sentences (4) and (5) have the same syntactic structure, the relevant subcategorization would necessarily be semantic in nature.

Schacter [1977], recognizing that semantics played some role in determining the acceptability of coordinate structures, proposed the following constraint:

> The Coordinate Structure Constraint (Schacter [1977])
>
> The constituents of a coordinate construction must belong to the same syntactic category and have the same semantic functions.

Thus, the reason sentence (5) is unacceptable is that the two conjoined constituents *slowly* and *with a Swiss army knife* do not have the same semantic functions; the former specifies the *manner* in which *he did the job* and the latter specifies the *instrument*.

We have already discussed some of the problems with the syntactic identity condition. The condition of identity of semantic function does seem to have some basis; however, the idea is not carried much beyond the initial intuition. Schacter suggests that "not everything that might be classified as a 'semantic function' is in fact relevant to the permissibility of coordinate conjunction, so that those functions that ARE relevant will have to be specified." For example, the semantic function of anaphoric reference is not relevvant to conjunction. In sentence (7) below, the constituent *he* has this semantic function while the constituent *Mary* to which it is conjoined does not.

(7) John said that he and Mary were just good friends.

Unfortunately, Schacter made no detailed attempt to specify what the relevant semantic functions are.

2. A Theory of Coordinate Constituent Conjunction

Given the evidence presented in Chapter 1, we can only conclude that not everything that is normally classified as syntactic nor everything that is normally classified as semantic is relevant to coordinate constituent conjunction (CCC), although some aspects of each are.

In the following sections, we will isolate the syntactic and semantic constraints on CCC. The result will be two necessary conditions, one dealing with syntax and the other with semantics, which together form a condition for CCC which is both necessary and sufficient. We begin with the syntactic constraint.

2.1 A Syntactic Constraint on CCC

Both Gazdar and Schacter (as well as other linguists) have suggested that for the conjunction of two constituents in a sentence to be grammatical, the constituents must belong to the same syntactic category. We pointed out some of the problems with such a constraint in the previous chapter. Rather than try to solve those problems by giving a specification for a grammar of English which would satisfy the constraint, we will simply avoid the problems by proposing a constraint that is more general — more general in the sense that it refers only to positions in the tree structure assigned to a sentence and not to the names of the nodes occupying the positions.

Informally, the proposal is as follows. For the conjunction of two constituents in a sentence to be grammatical, the two sentences which result from replacing the whole conjunction with each of the conjuncts (or their plural forms if they are subject noun phrases[1]), must be grammatical and have "compatible" syntactic analyses. This is formally stated as (C1).

(C1) A Syntactic Constraint on CCC

The sentence $S = X - \langle C_1 \text{ and } C_2 \rangle - Y$ is grammatical ONLY IF

$$S_1 = X - C_1' - Y \quad \text{and} \quad S_2 = X - C_2' - Y$$

where for i=1,2: if C_i is a subject NP then C_i' is the plural form of C_i, otherwise C_i' is C_i,

have acceptable syntactic analyses $A(S_1)$ and $A(S_2)$ which are identical except that the position filled by C_1' in $A(S_1)$ is filled by C_2' in $A(S_2)$.

It must be emphasized that (C1) is only a syntactic constraint. It does not state that the sentence S containing the conjunction is semantically equivalent to the conjunction of the resultant sentences S_1 and S_2. For example, the sentence

The man and woman danced together.

satisfies the constraint, but does not have the same meaning as

The men danced together and the women danced together.

Constraint (C1) correctly predicts that sentences (1) and (2) below are ungrammatical. In sentence (1), replacing the conjunction with the right-hand conjunct results in a sentence (1a) which does not have an acceptable syntactic analysis.[2]

(1) *By next week, Dave will *have quit his job* and *looking for a new one.*

(1a) *Dave will *looking for a new one.*

In sentence (2), replacing the conjunction with the right and left conjuncts result in sentences with syntactic analyses which are acceptable but which are not compatible.

(2) *Steve *wanted* and *promised* Peter to leave.

(2a) [$_S$ [$_{NP}$ Steve] [$_{VP}$ wanted [$_S$ [$_{NP}$ Peter] [$_{VP}$ to leave]]]].
(2b) [$_S$ [$_{NP}$ Steve] [$_{VP}$ promised [$_{NP}$ Peter] [$_S$ [t] [$_{VP}$ to leave]]]].

Note that sentences (3) and (4) below are acceptable. The corresponding sentences in which the conjunctions have been replaced by the left and right conjuncts are both acceptable and have compatible syntactic analyses.

(3) By next week, Dave will *have quit his job* and *be looking for a new one.*

(3a) By next week, Dave will [have quit his job].
(3b) By next week, Dave will [be looking for a new one].

(4) Steve *wanted* and *expected* Peter to leave.

(4a) Steve wanted [Peter to leave].
(4b) Steve expected [Peter to leave].

---

1. This is necessary to satisfy number agreement rules.

2. We are assuming that the two conjuncts are indeed constituents, an assumption some linguists might not make. Justification for our decision is given in section 2.4.1.

2.2 A Semantic Constraint on CCC

Constraint (C1) appears to be a necessary condition for the coordinate conjunction of two constituents but it is not a sufficient one. The following sentences in which the constituents in italics are assumed to be conjoined, satisfy the constraint but the conjunctions are not grammatical.

(5) *Chris gave the dog *a bone* and *to Margie.*

(5a) Chris [gave [the dog] [a bone]].
(5b) Chris [gave [the dog] [to Margie]].

(6) *Jim *knows* and *seems* a trustworthy person.

(6a) Jim knows [a trustworthy person].
(6b) Jim seems [a trustworthy person].

It is necessary to constrain the coordinate conjunction of two constituents on the basis of their semantic roles in the sentence as well as their syntactic positions.

The level of semantics which appears to be relevant to the coordinate conjunction is the one dealing with predicate/argument relationships between constituents in a sentence. Examples of predicate/argument relationships are the relationships between a verb or verbal noun and its cases and modifiers.

Before giving the details of our model of predicate/argument relationships (which can be found in section 2.5), we will present the constraint and give a few simple examples illustrating the correctness of its predictions.

(C2) A Semantic Constraint on CCC

The sentence $S = X - \langle C_1 \text{ and } C_2 \rangle - Y$ is grammatical ONLY IF

$$S_1 = X - C_1 - Y \quad \text{and} \quad S_2 = X - C_2 - Y$$

have acceptable predicate/argument analyses $P(S_1)$ and $P(S_2)$ which are are identical ignoring C_1, C_2, and their subconstituents; and the roles that are filled by C_1 in $P(S_1)$ are filled by C_2 in $P(S_2)$.

Consider again the ungrammatical sentences (5) and (6).

(5) *Chris gave the dog *a bone* and *to Margie.*

(5a) Chris gave the dog [a bone].
(5b) Chris gave the dog [to Margie].

(6) *Jim *knows* and *seems* a trustworthy person.

(6a) Jim knows [a trustworthy person].
(6b) Jim seems [a trustworthy person].

In (5a) the constituent *a bone* is related to *gave* as the verb's neutral (or patient) case. In (5b) the constituent *to Margie* is related to *gave* as the verb's dative case. Given these relationships, constraint (C2) correctly predicts that (5) is ungrammatical. The constraint also predicts that (6) is ungrammatical. In (6a), the constituent *a trustworthy person* is related to *knows* as the verb's neutral case. The verb *seems* in (6b) is used copulatively; the constituent *a trustworthy person* specifies an attribute of the NP *Jim*.

Note that the following sentences in which the right conjuncts in (5) and (6) have been suitably replaced, are grammatical.

(7) Chris gave the dog a bone and some biscuits.
(8) Jim seems and is a trustworthy person.

2.3 A Necessary and Sufficient Condition for CCC

In the previous two sections we proposed two constraints, (C1) and (C2), and presented a number of ungrammatical sentences as evidence of their validity (sentences (1) and (2) which violated (C1); and (5) and (6) which violated (C2)). We also presented a number of grammatical sentences which satisfied both (C1) and (C2) (sentences (3), (4), (7), and (8)). Given this evidence, we propose that constraints (C1) and (C2) taken together provide a condition on coordinate constituent conjunction which is both necessary and sufficient.

(C3) A Necessary and Sufficient Condition for CCC

The sentence $S = X - <C_1 \text{ and } C_2> - Y$ is grammatical IF and ONLY IF for

$$S_1 = X - C_1' - Y \text{ and } S_2 = X - C_2' - Y$$

both (i) and (ii) hold:

(i) S_1 and S_2 have acceptable syntactic analyses $A(S_1)$ and $A(S_2)$ which are identical except that the position filled by C_1' in $A(S_1)$ is filled by C_2' in $A(S_2)$.

(ii) S_1 and S_2 have acceptable predicate/argument analyses $P(S_1)$ and $P(S_2)$ which are identical ignoring C_1', C_2', and their subconstituents; and the roles that are filled by C_1' in $P(S_1)$ are filled by C_2' in $P(S_2)$.

It is important to note that this proposal would not be as appealing if the rules governing the syntactic analysis of a sentence and the assignment of predicate/argument relationships to the constituents were specially tailored to fit the claim. However, this is not the case; we have chosen standard syntactic and predicate/argument models that are widely accepted by linguists. The following sections will be devoted to providing more detailed descriptions of these models.

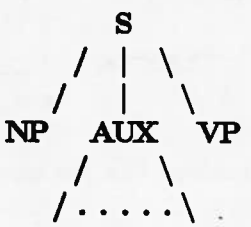
2.4 A Model of Syntax

The conjunction condition refers only to syntactic analyses (or surface structures) and not to the rules deriving the analysis. Therefore this section will describe the syntactic structures we are assuming and will not be concerned with the format of the rules. Furthermore, since there is fairly good agreement among linguists on what these structures are, we will discuss only those over which there might be some disagreement.

2.4.1 VP Structure

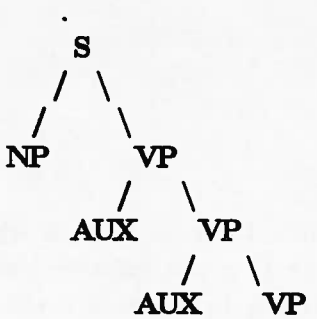
Many grammars of English assume that the auxiliaries preceding a main verb form a single constituent which is a subconstituent of only the immediate sentence constituent. This is illustrated by the tree structure in (9).

(9)



In our model, an auxiliary and the VP following it form another VP constituent, resulting in structures like the one in (10).

(10)



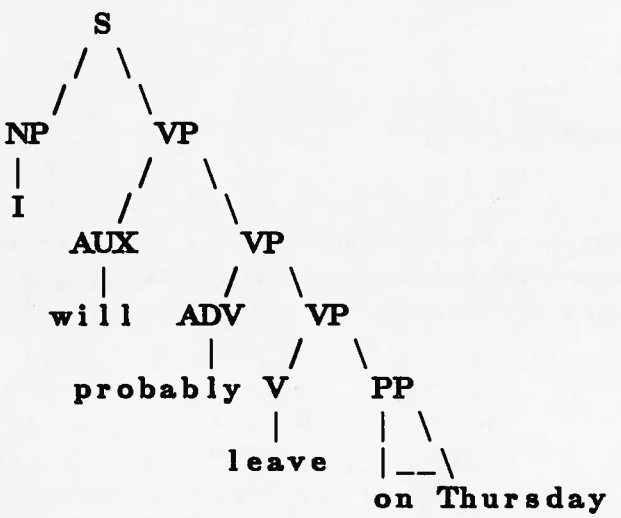
This structure was proposed by Sag [1976] to explain VP deletion phenomena and was justified on other grounds as well. The structure allows for a simple account of conjunctions in which an auxiliary begins the right conjunct, as in sentence (11).

(11) Jack left today and will return tomorrow.

The conjunction *and* directly conjoins the two VPs, *left today* and *will return tomorrow.*

Modifiers such as time and place modifiers which follow the main verb are directly dominated by the main VP (the one directly dominating the main verb), not by the S. A modifier preceding the main verb (but not the subject) forms a VP constituent with the VP following it. This is illustrated in (12).

(12) I will probably leave on Thursday.



This allows for a simple account of conjunctions like the ones in (13a) and (13b).

(13a) I will probably leave on Thursday and return on Sunday.
(13b) I will leave on Thursday and probably return on Sunday.

2.4.2 NP Structure

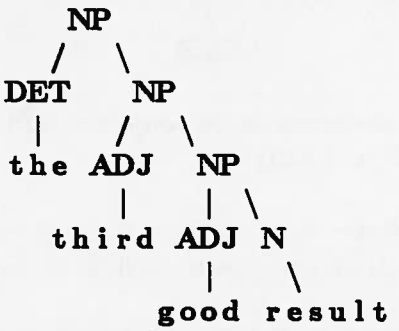
Many grammars assign structures to NPs in which the determiner and the adjectives preceding the head noun, and the PPs and relative clauses following it are at the same level. Our model assumes structures which better reflect the way in which the individual elements function. For example, consider sentences (14a) and (14b).

(14a) That was the third good result.
(14b) That was the third best result.

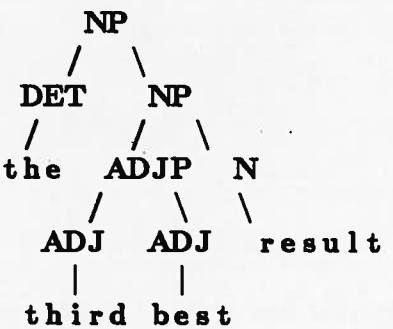
In (14a), the adjective *third* modifies *good result*, and *good* modifies *result*; *third* and *good* do not

independently modify *result*. In (14b), *third* modifies *best*, and the phrase *third best* modifies *result*. In both sentences *the* functions as a modifier making the subphrases *third good result* and *third best result* definite. The constituent decomposition of the NPs reflecting these functions are given in (15a) and (15b).

(15a) the third good result

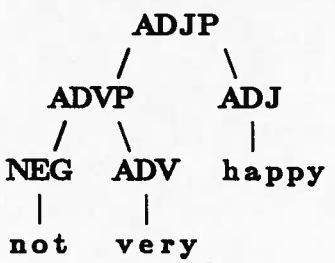


(15b) the third best result



Similarly, adverbs modifying adjectives will be appropriately grouped as shown in example (16).

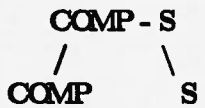
(16) not very happy



2.4.3 COMP-S Structure

The structure of a complement sentence which is marked by *that* or *for* is given in (17).

(17)



This allows two sentences to be conjoined and marked by the same complement marker as in the sentences in (18) and (19).

(18) I think that Peter will drive and Sue will walk.
(19) For Peter to drive and Sue to walk is ridiculous.

Note that we are assuming that *Peter to drive* and *Sue to walk* in (19) form S constituents. This assumption also holds when such a string follows a verb like *want* as in (20).

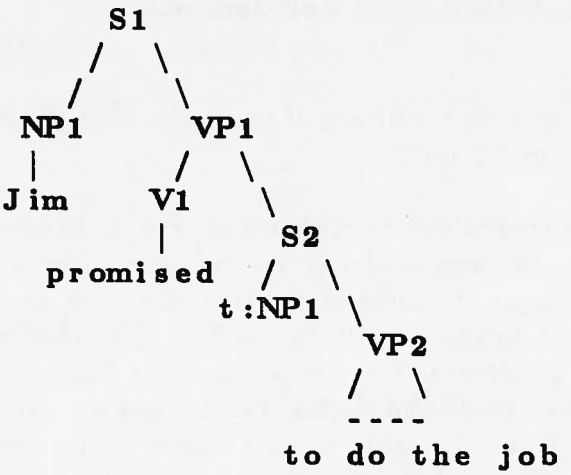
(20) I want Sue to walk and Peter to drive.

2.4.4 Traces

Finally, if a constituent has more than one grammatical role, the syntactic structure representing the sentence will include a *trace* of the constituent in the positions corresponding to each of its secondary roles. For example the surface structure of of sentence (21) is taken to be as shown in (22).

(21) Jim promised to do the job.

(22)



The structure in (22) explicitly represents the fact that NP1 is the subject of VP2.

2.5 A Model of Predicate/Argument Relationships

In this section we provide the details of the model of predicate/argument relationships to which we referred in sections 2.2 and 2.3.

In our model, the predicates of a sentence are the verbs (or words whose root forms are verbs; e.g. gerunds) and the prepositions in the sentence. Any constituent which is directly dependent on a predicate is an argument of the predicate. The model describes the possible relationships between predicates and arguments and how these relationships are determined from a constituent analysis of a sentence. Its motivation comes from the work done on verb case relationships by Schacter, Stockwell, and Partee [1973]. Thus we will begin by describing a simple case grammar and a set of rules for mapping case frames to grammatical roles devised by SS&P and then present the general model of predicate/argument relationships.

2.5.1 A Simple Case Grammar for English Verbs

In the case grammar each verb is assumed to take a subset of the following five cases, where the name of the case specifies its relationship with the verb.

neutral - Chris bought *a windsurfer.*

dative - Kathy gave a book *to Margie.*

locative - Joyce put the flowers *in a vase.*

instrument - Bill fixed the faucet *with a wrench.*

agent - *Janet* sang an aria.

Each verb has a case frame specifying the set of cases taken by the verb and for each case, whether it is optional or obligatory. For example, the case frame for the verbs *give* and *sing* would be:

give: neutral dative agent
sing: (neutral) agent

The presence of brackets around a case indicate that the case is obligatory; otherwise it is optional.

SS&P imposed the order:

neutral, dative, locative, instrument, agent

on the five cases and formulated the following three rules for mapping the cases of a verb's case frame to the grammatical relations of subject, object, and prepositional phrase of an active sentence.

R1. Finding the Subject: the rightmost case in the ordering if it is obligatory must become the subject. If it is optional, it may be discarded and the rule applied to the remaining cases.

R2. Finding the Objects: the objects are found by reading from left to right until the number of objects is used up. The objects occur with no preposition.

R3. Prepositional Phrases: the remaining cases occur marked by prepositions. Each case has a default marking preposition associated with it. If a verb requires some other preposition, it must be specified in the verb's case frame.[1]

To illustrate how these rules work consider the ordered case frame for the verb *break*:

break: neutral (instrument) (agent)

Since both the agent and the instrument cases are optional, both could be discarded by successive applications of R1, leaving only the obligatory neutral case to become the subject. This gives the structure in (23), illustrated by sentence (23a).

(23) NEUTRAL break
(23a) The rock broke.

The three other possible structures are given below.

(24) INSTRUMENT break NEUTRAL
(24a) The rock broke the window.

(25) AGENT break NEUTRAL
(25a) Jack broke the window.

(26) AGENT break NEUTRAL INSTRUMENT
(26a) Jack broke the window with a rock.

SS&P also formulated rules that allow case frames to be reordered. This was to handle such verbs as *give* whose dative can appear unmarked by a preposition before the neutral, as in: *Joe gave Tom the book*; and *smear*, whose locative can appear as object before the neutral which

---

1. It is possible for a case to be marked by more than one preposition. For example the locative case of the verb *put* can be marked by any one of the following prepositions:

on, onto, in, into, under, above, ...

is marked by *with*, as in: *We smeared the wall with paint* (vs. *We smeared paint on the wall*). [2]

An inverse mapping from grammatical roles (subject, object, prepositional phrase) to case roles can be obtained by specifying in the case frame the semantic features a constituent must have in order to fill a particular case and including a test for these features in the mapping rules. To illustrate the necessity of such semantic tests, consider the following sentences.

(27) Juan attached the wire with a pair of pliers.
(28) Juan attached the wire with great care.

Both sentences have the grammatical structure: SUBJECT attached OBJECT PP[with]. However, in (27) the PP *with a pair of pliers* fulfills the instrument case of the verb *attached*; in (28) the PP *with great care* does not fulfill any case role according to the case grammar of SS&P. This distinction can be made only by performing a semantic test on the NP of each of the PPs.

It is important to note that the sample case frames and the mapping rules described above are concerned only with verbs in their active finite forms. In general, the case frames and the default prepositions for a verb case are different for the different forms of a verb. For example, the case frames for the passive, active, gerund, infinitive, and complement forms of the verb *sing*, are almost all different. The five case frames are given below.

sing-active: (neutral) agent
sing-passive: (agent) neutral
sing-gerund: (agent) (neutral)
sing-infinitive: (neutral)
sing-complement: (neutral) agent

For the active, infinitive, and complement forms of a verb neither the agent case nor the neutral case have default marking prepositions; for the passive and gerund forms, the agent has the default marking preposition *by*; and for the gerund form, the neutral has the default marking preposition *of*. Rules R1-R3 may be applied to the case frames for both the active and passive forms of a verb to produce active and passive structures respectively. The structures shown in (29) result from the application of the rules to the sing-active frame and the structures shown in (30) result from their application to the sing-passive frame.

(29) AGENT sing-active (NEUTRAL).
(29a) Kathy sang (the song).

(30) NEUTRAL sing-passive (AGENT).
(30a) The song was sung (by Kathy).

---

2. There are alternative ways of handling this problem; for instance, by assigning more than one case frame to a verb.

Not only do the gerund forms of verbs have different case frames and different default marking prepositions for cases, but the rules R1-R3 do not apply to map the frames into grammatical structures. Therefore other rules are needed to form gerund structures like those shown in (31) and (32) below.

(31) sing-gerund (NEUTRAL) (AGENT)
(31a) The singing (of the song) (by Kathy) ...

(32) AGENT sing-gerund (NEUTRAL)
(32a) Kathy's singing (of the song) ...

Rules are also needed to form infinitive and complement structures shown in (33) and (34) below.

(33) AGENT sing-complement (NEUTRAL)
(33a) Kathy wanted (t) to sing (the song).
(33b) ... wanted Kathy to sing (the song).

(34) sing-infinitive (NEUTRAL)
(34a) To sing (the song) ...

Note that in (33a) the constituent *Kathy* functions as the agent of both verbs *wanted* and *sing*. Sentence (35) demonstrates that a single constituent can also function as different cases of two verbs.

(35) Brock used the wrench to tighten the screw.

The constituent *the wrench* functions as the neutral of *used* and the instrument of *tighten* (*Brock* functions as the agent of both these verbs). Although it is possible for a single constituent to function as cases of different verbs, it is not possible for a single constituent to function as two different cases of the same verb. For example, a single constituent cannot fill both the neutral and dative cases of the verb *give*. It is also not possible for two unconjoined constituents having equal rank to function as the same case of a verb.[3] Therefore, not only must two constituents fill the same case to be conjoined, but for two constituents (of equal rank) to fill the same case they must be conjoined.

---

3. In sentences like the following one (due to Bresnan [1982]) the two constituents fulfilling the location case of the verb *hid* do not have equal rank.

John hid in the garden behind the trees.

The constituent *behind the trees* is a "further specification" of *in the garden*. Note that the above sentence does not have the same meaning as the one below.

John hid in the garden and behind the trees.

2.5.2 A Model for General Verb/Argument Relationships

It was mentioned above that in our model the constituents which function as predicates are the verbs and prepositions in a sentence. Any other constituent functions as an argument of one of these explicit predicates or as an argument of a dummy predicate. (Dummy predicates are discussed in section 2.5.4.) Given this viewpoint, the simple case model described above is inadequate to capture all the possible verb/argument relationships in a sentence. Thus we add to the five case arguments of SS&P the following list of arguments.

manner: Paul did the job *with great care.*

location: Duncan plays squash *at the University.*

direction: Glen sailed *away from the shore.*

source: The water comes *from a natural spring.*

time:[4] Rod will arrive *on Tuesday.*

duration: Chris worked on his thesis *for two years.*

frequency: Barbara runs *twice a week.*

cardinality: Wendy has been to Europe *five times.*

ordinality: Peter went to Mexico *for the first time.*

degree: I *very much* regretted the misunderstanding.

probability: Sue will *likely* be at the game.

reason: Mary went to the store *to buy some milk.*

accompaniment: Grant played squash *with Andrew.*

attribute: Chris is a *good person.*

predicate-phrase: Mark has *been cycling.*

---

4. Actually several time arguments are needed to differentiate between the different degrees to which the time of a predicate can be specified. For example, in the sentence:

She is coming today at two o'clock.

at two o'clock specifies a precise time ("clock time"), whereas *today* specifies a 24-hour time range ("day time").

The last two are non-standard arguments which have been included to allow for a general treatment of verb/argument relationships. All verbs, including the auxiliary and copular verbs, are considered to be predicates and any constituent related to a verb functions as its argument. This allows us to account for ungrammatical sentences such as the sentence (36).

(36) *Chris is *going to the store* and *a good person.*

The constituents *going to the store* and *a good person* cannot be conjoined because, in the derived sentences (36a) and (36b), they have different relationships with the verb *is*.

(36a) Chris is going to the store.

The phrase *going to the store* functions as a predicate-phrase argument of the verb *is*.

(36b) Chris is a good person.

The phrase *a good person* functions as an attribute argument of the verb *is*.

Furthermore, the model allows us to account for the fact that (37) is grammatical while (38) is not (or is at best peculiar).

(37) Jack's car is small and yellow.

(38) ?Jack has a small and yellow car.

If adjectives and adjective phrases were treated as predicates the model could not account for both (37) and (38).

In this general model, "argument frames" replace the case frames of the case model. The argument frame for each verb will contain the set of arguments taken by the verb and information about each argument (i.e., the case information described above for the case arguments and similar information for the other arguments).

Extending the rules which map grammatical roles to case roles so that they map grammatical roles to general argument roles should not be difficult as most of the above arguments can be freely ordered (the constituents which fulfill them usually contain sufficient semantic information for their roles to be determined).

2.5.3 Preposition/Argument Relationships

The argument frames for prepositions are much simpler than for verbs since every preposition takes one argument: an object. For each sense of a particular preposition there will be an argument frame describing the semantic properties that a constituent must have to fulfill the object role. For example, the argument frame of the preposition *with* having the sense of "using as instrument", would specify that the object role must be fulfilled by a constituent

which refers to a physical object.[5] In sentence (39) the constituent in italics fulfills the object role of the preposition *with* having the above sense.

(39) I hit the nail with *a hammer*.

The preposition *with* can also have the sense of "using as manner" as in sentence (40).

(40) I hit the nail with *abandon*.

In this case, the constituent filling the object role must have the semantic property that it refers to a manner.

Note that sentence (41) in which the constituents *a hammer* and *abandon* have been conjoined as joint object of *with* is unacceptable.

(41) *I hit the nail with *a hammer* and *abandon*.

2.5.4 Dummy Predicates

Dummy predicates are used to express the relationships between constituents which function as arguments of a predicate which is not explicit in the sentence. For example, consider sentence (42) below.

(42) Jack owns a small yellow car.

The adjectives *small* and *yellow* specify particular properties of the noun *car*, namely size and colour respectively. The dummy predicate *has-physical-properties* is used to relate these three constituents. Its arguments are an obligatory subject, and the optional arguments size, colour, rank, quantity, etc. A constituent functioning as the subject must have semantic features which describe a physical object and the constituents functioning as the other arguments must have appropriate semantic features.

The reason for this approach rather than one in which noun modifiers themselves are treated as predicates is to allow the different relationships that modifiers (adjectives, PPs, etc.) may have with nouns to be made explicit. This is important for conjunction as the conjunction condition states that only constituents which can have the same relationships with other consti-

5. Note that in English we often must deduce the sense of the preposition by examining the semantic properties of the possible objects of the preposition and the semantic properties of constituents to which the prepositional phrase may be related. Hirst and Charniak [1982] and Hirst [1983] describe a system that does this.

tuents in a sentence may be conjoined. It would not be possible to express these different noun/modifier relationships if modifiers were treated as predicates in our model as the relationship between a predicate and an argument (and hence between two arguments of a common predicate) is expressed by the name of the argument. For example, if both *yellow* and *small* were treated as predicates taking a single subject argument, there would be no way to account for the unacceptability of sentence (43) below.

(43) *Jack owns a small and yellow car.

Dummy predicates are also used to express the relationships between an adjective (or ADJP) and a modifying adverb (or ADVP) as in the phrases *unusually efficient* and *not very large*; and between an adverb and another modifying adverb as in, *not very*. They are also used to describe the function of a major S. There are three such predicates taking a major S as an argument: *is-declarative*, *is-imperative*, and *is-question*.

2.5.5 Conjunctions

Because we have been concerned with predicate/argument relationships as a basis upon which to determine the acceptability of conjunctions, we have ignored the relationships between a conjunction and the constituents it conjoins. To complete the model, the coordinate conjunction *and* will be taken to be a predicate taking two obligatory arguments: a left conjunct and a right conjunct. There are no semantic properties associated with the arguments of a conjunction since any two constituents may be conjoined as long as they satisfy the conjunction condition. In other words, the semantic properties two constituents must have to function as arguments of a conjunction depend on the predicate/argument function fulfilled by the constituent resulting from their conjunction.

2.6 Summary

In this chapter we proposed a necessary and sufficient condition for coordinate constituent conjunction, which we repeat below.

(C3) A Necessary and Sufficient Condition for CCC

The sentence $S = X - \langle C_1 \text{ and } C_2 \rangle - Y$ is grammatical IF and ONLY IF for

$$S_1 = X - C_1' - Y \quad \text{and} \quad S_2 = X - C_2' - Y$$

both (i) and (ii) hold:

(i) S_1 and S_2 have acceptable syntactic analyses $A(S_1)$ and $A(S_2)$ which are identical except that the position filled by C_1' in $A(S_1)$ is filled by C_2' in $A(S_2)$.

(ii) S_1 and S_2 have acceptable predicate/argument analyses $P(S_1)$ and $P(S_2)$ which are identical ignoring C_1', C_2', and their subconstituents; and the roles that are filled by C_1' in $P(S_1)$ are filled by C_2' in $P(S_2)$.

To illustrate that (C3) correctly predicts the acceptability of coordinate structures given the models presented in 2.4 and 2.5, we present the following collection of sentences. The first ten sentences are ungrammatical; (1)-(5) violate part (i) of the condition and (6)-(10) violate part (ii). The last ten are grammatical sentences which satisfy both constraints.

(1) *The *tired* and *in a foul mood child* was sent to bed.

(2) *Jim went *San Francisco* and *to Los Angeles.*

(3) *The Expos *beat the Jays* and *beaten by the Giants.*

(4) *I was *tired* and *practically to fall asleep on my feet.*

(5) *The piano which Mark *owns* and *plays a guitar* is out of tune.

(6) *Firth broke *the radio* and *into the house* with a hammer.

(7) *The *third* and *difficult* problem has been solved.

(8) *Rod *probably* and *skillfully* drove the car.

(9) *The girl with *Joe* and *blonde hair* is my sister.

(10) *Chris *wanted* and *appeared* to have a good time.

(11) The tired and cranky child was sent to bed.

(12) Jim went to San Francisco and to Los Angeles.

(13) The Expos beat the Jays and were beaten by the Giants.

(14) I was tired and practically falling asleep on my feet.

(15) The piano which Mark owns and plays is out of tune.

(16) Ian broke the radio and the television with a hammer.

(17) The third and last problem has been solved.

(18) John carefully and skillfully drove the car.

(19) The girl with blond hair and blue eyes is my sister.

(20) Chris wanted and expected to have a good time.

PART II

3. Parsing Coordinate Constituent Conjunctions

In this chapter we illustrate how the results of the last chapter can be applied to the problem of parsing sentences containing coordinating conjunctions. We will begin by describing the parsing system that has been chosen for this illustration.

3.1 The Parsing System

The parsing system has two components: the first produces a syntactic analysis of the input sentence, and the second produces a predicate/argument analysis based on the syntactic analysis. These two components are modelled after the parser and the case frame builder developed by Marcus [1980].

3.1.1 The Parser

Although the parser we hypothesize will have the same structure and adhere to the same basic principles as Marcus's parser, it will be necessary to extend and modify it to suit our purposes. Changes will be made to both the grammar and the grammar interpreter. The changes to the grammar will include the addition of new grammar rules specifically for handling conjunction, and the replacement or modification of some of Marcus's rules which do not produce structures according to our model of syntax. (The grammar rules necessary to parse the example sentences in this chapter are given in Appendix A.) Any changes to the grammar interpreter are specifically for handling conjunction. They will be introduced along with the new conjunction rules with the proper motivation in section 3.2. What follows is a description of the parser developed by Marcus [1980].

The main characteristics of Marcus's parser are these:

1. The parser operates strictly deterministically, in the sense that any structure that is built by the parser is permanent. It has no general backtracking facility.
2. The parser has limited access to contextual information.

The parser consists of a grammar expressed as a set of pattern/action rules and a grammar interpreter which selects and executes the rules of the grammar. The output of the parser is a tree representing the constituent structure of the input sentence. Each node in a complete tree has four components: a type, a set of grammatical features, a parent, and a list of daughters. Each node is also assigned a unique name composed of its type and a number (e.g.,

NP15, AUX33, WORD5). The type of a node is one of the usual syntactic categories (S, NP, VP, etc.). The grammatical features of a node give the parser information about the constituent the node represents which is not inherent in the node's type. (Actually, the type of a node is included in the node's the feature set.) This information is typically necessary to determine the grammatical role of the node or the effect the node has on the grammatical behavior of nodes which dominate it. For example, when the parser is analysing a verb phrase it must know what kinds of complements the head verb can take, so features supplying this information are included in the feature set of each verb in the lexicon.

The operation of the parser is neither strictly bottom-up (data driven) nor strictly top-down (hypothesis driven), but a combination of the two. For this reason two data structures are maintained by the parser: a push-down stack which is a natural data structure for the top-down aspects of the parser, and a buffer of contiguous cells which is a natural structure for the bottom-up aspects of the parser.

The stack, called the "active node stack" is used as storage for incomplete constituents; that is, constituents that the parser is still actively constructing. The parser initiates a constituent by creating a new node and pushing it onto the stack. It then attempts to attach to the new node already-constructed constituents, which are stored in the buffer. This new node may be covered up by other nodes before it is complete while the parser builds the lower-level constituents which will be its daughters. When a node is completed, it is popped from the stack.

At any given time only two of the nodes in the stack can be directly examined or modified. These are the bottom-most node (the stack grows downward), called the "current active node" or "**c**", and the cyclic node (NP or S) closest to the bottom of the stack, called the "current cyclic node".

The buffer is a linear list of five cells X(1),...,X(5) each of which can be either occupied by a grammatical constituent which has been popped from the stack or by a word from the input sentence which has undergone morphological processing. The buffer has the property that all non-empty cells are contiguous starting at the first cell. Although the buffer has five cells, the grammar interpreter has access to only three contiguous cells while executing a given rule. That is, it has a sliding three cell window into the buffer. The first accessible buffer cell is X(i), where i is the current buffer offset. The buffer offset (i) must not exceed the length of the buffer. (Marcus claims that in practice i is never greater than 3.) The buffer offset is initially set to 1 and is changed before the execution of special rules, which will be described below. The three accessible cells at any given time, X(i), X(i+1), and X(i+2) will be referred to by C(1), C(2), and C(3), and their contents by "1st", "2nd", and "3rd", respectively.

We mentioned above that the grammar interpreter builds constituent structures by selecting and executing rules of the grammar. The form of these rules and the method by which rules are selected by the grammar interpreter are described below.

Each of the rules in Marcus's grammar has two components: a pattern and an action. A pattern consists of a list of predicates which test a subset of the five accessible nodes (the three buffer nodes and two stack nodes) and their children for the presence or absence of features. An action is a small program which specifies the operations the parser is to perform on these nodes if the pattern of the rule is satisfied. (Marcus says that a pattern "matches" the subset of nodes if the predicates of the pattern are satisfied by them.)

Each rule is assigned a numerical priority which the grammar interpreter uses to decide which pattern to test first when two or more patterns potentially match. The priorities are non-negative integers and the highest priority is 0.

The rules are organized into packets which can be activated and deactivated by rule commands. Each node in the active node stack has a set of packets associated with it which were active when the node was the current active node. These packets contain the rules which are relevant to the grammatical environment defined by the node (i.e., the rules which apply either to attach daughters to the node or to create the nodes which will be its daughters). Thus, only the packets which are associated with the current active node are active.

At each point in the parsing process, the grammar interpreter tests the patterns of the active rules (the rules in active packets) in order of descending priority and executes the action of the first rule whose pattern is satisfied.

Some of the rules in the grammar are distinguished as "attention shifting" (or "AS") rules. They are so called because they cause the grammar interpreter to shift its attention away from the pattern it is currently testing. This is done in the following way. Before the ith predicate of a pattern is applied to the node in C(i) the grammar interpreter checks to see if there is an AS rule whose pattern is satisfied with a buffer window starting at C(i). If so, the buffer offset is set to i and the window slides to display C(i), C(i+1), and C(i+2). The parse will continue with this offset until attention is shifted back by the execution of the rule command "restore the buffer". At this point the grammar interpreter will continue the pattern match that was interrupted by the AS rule.[1]

The grammar rules are written in a language called PIDGIN. Each rule in PIDGIN is of this general form:

{(AS) Rule <name> (Priority: <numerical priority>)
in <Packet list>
<pattern> --> <action>}

Patterns are written in PIDGIN as a list of descriptions, each enclosed in square brackets. Each description contains a list of predicates and refers to one of the nodes in the buffer, to the

1. Attention shifting rules are typically used in this grammar to start the parse of noun phrases and the buffer restore command is executed by the rule which pops the noun phrase node from the stack.

current active node, or to the current cyclic node. A description which refers to either of the latter two nodes begins with the label "**c". Otherwise the node to which a description refers is given by its position in the pattern. That is, the first description not beginning with **c refers to 1st (the constituent in the first buffer cell); the second description, to 2nd; and the third description, to 3rd. Some patterns from Marcus's grammar rules and their meanings are given in figure 1.

[=np] [=verb]
1st is a noun phrase (i.e., has the feature np).
2nd is a verb.

[=pp] [* is any of prep, adj; * is not part]
1st is a prepositional phrase.
2nd is a preposition or an adjective and is not a participle.

[=pp] [**c; the node above the np above * is vp]
1st is a prepositional phrase.
The node above the current cyclic node (which in this case is the NP above the current active node) is a verb phrase.[2]

Figure 1. Grammar rule patterns.

PIDGIN also has the predicate "t", which is satisfied by any node. The description [t] should appear in a pattern only if the action of the rule is independent of the node to which [t] corresponds. (Marcus sometimes uses [t] in a rule pattern to force a word into the buffer so that a test can be performed on it in the action of the rule. These tests do not account for the possibility that the word is a conjunction.)

Actions are written in PIDGIN as lists of commands. The basic action commands operate on nodes. There are PIDGIN primitives to create new parse nodes, attach nodes in the buffer to the current active node, and pop the current active node from the stack. PIDGIN also provides primitives to insert a specific lexical item into a specific buffer cell (although it is not

---

2. Here Marcus uses the operator 'above' as a general tree climbing operator. Formally, the operator is allowed to be used only to access the current cyclic node, as in the PIDGIN phrases: the S above *, the NP above *. The action following this particular pattern is a semantic test to determine where the PP should be attached. Marcus states, "These tests, which fall outside of the range of the research described in the rest of this document, seem to need to scan most of the active node stack. In many ways, PP attachment seems to be different from the sorts of phenomena discussed in this document" (Marcus [1980] p. 261).

clear that this is either desirable or necessary), to assign features to one of the accessible nodes, and to activate and deactivate packets of rules.

Primitives allowing conditional expressions are provided by PIDGIN, but no control primitives allowing iteration within rules nor calls to subroutines are provided.

An example of a rule from Marcus's grammar is given below.

```
{RULE THAT-S-START PRIORITY:5 IN CPOOL
[=comp,*that] [=np] [=verb] -->
Label a new s node sec, comp-s, that-s.
Attach 1st to c as comp.
Attach 2nd to c as np.
Activate cpool, parse-aux.}
```

This rule recognizes the beginning of a "that-S complement" as in sentence (1).

(1) He believes that the earth is round.

When the rule is executed, a new S node is created and pushed onto the stack thus becoming the current active node. This node is then labelled with the features SEC (secondary or non-major), COMP-S (complement), and THAT-S (embedded finite complement). The first node in the buffer is attached to **c** and assigned the type COMP. The second node is attached to **c** to the right of the first node and is assigned the type NP. The packet CPOOL, which contains rules to parse clause-level constituents, and the packet PARSE-AUX, which contains rules to parse auxiliaries, are activated and associated with the current active node. Given sentence (1) as input, the parser would be in the state depicted in figure 2 just after THAT-S-START is executed.

Rule about to run: STARTAUX in PARSE-AUX

Active Node Stack:

.
.

c: S2 (SEC COMP-S THAT-S S) / (CPOOL PARSE-AUX)
COMP: (that)
NP: (the earth)

Buffer:

1: WORD6 (*be, verb, auxverb, ..): is

Figure 2. State of the parser after executing THAT-S-START

3.1.2 The Predicate/Argument Analyser

The predicate/argument analyser is taken almost directly from Marcus's case frame builder (CFB). Like the CFB, it is a separate process which constructs a predicate/argument representation of the input sentence from the syntactic analysis of the parser. The representation created by the CFB consists of "case frames" (not to be confused with the verb case specifications also called case frames) which are associated with nodes in the parse tree which function as predicates. A case frame consists of four components each containing one or more "slots":[3]

1. Predicate: the root of the word whose case frame it is.

2. Specifiers: added information about the predicate, such as the auxiliaries preceding a verb or the determiner preceding a noun.

3. Cases: the filled cases present in this use of the predicate.

4. Modifiers: phrases which are case frames themselves used to modify an entire case frame

3. The following is taken directly from Marcus [1980].

rather than to specify a case. Modifiers are optional sentence level comments such as time or location.

Our predicate/argument model, although different in many ways from that used by Marcus, can also be represented with frames, which we will call "predicate/argument frames". Since in our model every constituent functions either as a predicate or an argument, our frames will have only two components:

1. Predicate: the node dominating the constituent functioning as a predicate; or a dummy predicate.

2. Arguments: the nodes dominating the constituents functioning as arguments of the predicate.

Given the similarity between our predicate/argument model and Marcus's model, the method used by the case frame builder, which we describe below, can be used to map the syntactic analysis produced by the parser into a frame representation based on our predicate/argument model.

The case frame builder in Marcus's system runs in parallel with the parser by means of a set of programs that monitor the parser's progress. (The monitor programs for our grammar are given in Appendix B.) Each program includes a clause describing the parser action which is to activate the program code. These clauses can describe two parser actions: the creation of a node of a given type and the attachment of a node of a given type to another node of a given type.

The body of a monitor program can specify that the CFB create a new frame or fill a slot in a frame with a constituent having a particular grammatical role. For example, one monitor will detect the attachment of a verb to a VP. It will tell the CFB to create a new frame, to insert the verb into the predicate slot, and if the current S dominates a "bound" NP, to insert the NP into a case slot that can be filled by a constituent having the grammatical role subject.

Although there is only one way for the set of constituents which function as cases of a predicate in an unambiguous sentence to fill the case slots in the predicate's case frame, there is in general more than one way in which a proper subset of the set can fill the slots. Therefore as the CFB is given constituents for a frame, it must keep track of the various ways in which the frame can be filled. It does this in the following way. When the CFB receives the first constituent for a frame, it creates a list of hypotheses giving the different cases the constituent can fill, and for each case, which ones remain to be filled. Each time the CFB receives a new constituent to fill a slot in the frame it will update the frame hypotheses, discarding any which are no longer possible given the new constituent. For example, if the case frame builder is told that the verb *break* is to fill a predicate slot of a new frame and that the NP *the machine* having the grammatical role subject is to fill a case slot in this frame, two hypotheses would be generated.

The first would describe the possibility that *the machine* fills the neutral slot and no slots remain to be filled as in *The machine broke.* The second would describe the possibility that *the machine* fills the agent slot and the obligatory neutral slot remains to be filled as in *The machine broke the rocks.* If the next constituent supplied to the CFB for that frame is the NP *the rocks*, having the grammatical role object, the first hypothesis would be discarded and the second one appropriately updated.

In order to determine whether or not a constituent, given its grammatical role, can fill a particular case slot of a frame, a semantic test must be performed. This test involves comparing a set of semantic markers that are associated with the case slot with another set that are associated with the constituent in question. The semantic markers used for these tests are based upon the semantic markers of Fodor and Katz [1964]. The comparison is a simple intersection of the two sets: if the intersection is non-empty then the constituent can fill the slot, otherwise it cannot.

The CFB can also determine how well a constituent fills a particular slot. Thus it can perform simple comparative tests to determine the semantic preference for one constituent over another in a particular slot, or the preference for one slot over another slot for a particular constituent. This is done by partitioning the set of semantic markers into two classes: "good" markers and "ok" markers. By intersecting the markers associated with a constituent with each of these subsets individually, the CFB can conclude that the "fit" of the constituent in the slot is "good" (if the intersection with the "good" subset is non-null); or else "ok" (if the intersection with the "ok" subset is non-null); and otherwise "bad".[4]

To determine how well a constituent consisting of two coordinately conjoined constituents fills a slot, the semantic markers associated with each conjunct would first be intersected and then the resulting set intersected with the markers associated with the slot. Thus, the fit of the conjunction is "good" if the fits of both conjuncts are "good"; "ok", if one of the fits is "ok" and the other is either "ok" or "good"; and "bad", if at least one of the fits is "bad".

3.1.3 Interaction Between the Parser and the Predicate/Argument Analyser

For the most part, the operation of the parser is independent of the operation of the predicate/argument analyser. However because the parser is deterministic, a certain amount of interaction is required. The major reason is to allow the parser to ensure that the attachment of a constituent does not make a valid predicate/argument analysis impossible. For example, before attaching an object NP to a VP node it would ask the predicate/argument analyser if the NP can fill a slot in the frame belonging to the verb dominated by the VP. If it can, then it is

4. A better way of determining how well a constituent fills a case slot is described by Hirst [1983].

attached, otherwise the NP is left in the buffer to be attached later to some higher VP node, if there is one.[5]

The parser must also be able to determine that the conjunction of two nodes N_1 and N_2 would not make it impossible for the predicate/argument constraint of the conjunction condition to be satisfied before conjoining the nodes; that is, to determine, given one of the nodes, say N_1, occurs in a set of predicate/argument frame hypotheses, that N_2 is able to fill at least one of the hypo-slots filled by N_1.

5. In Marcus's grammar such tests are performed only if the constituent could be attached elsewhere in the tree. For example, it does not perform a test before attaching an object NP to a major VP node, since there would be no VP node higher in the tree to which it could be otherwise attached. In our grammar such tests will always be performed if the parser knows that the constituent to be attached to the current active node is to fill some slot in the argument frame associated with it or one of its subconstituents. For this reason, special rules to handle embedded clauses and verb phrases are not necessary.

3.2 A Method of Parsing Coordinate Constituent Conjunctions

In this section we describe a method of parsing sentences containing coordinate conjunctions using the predicate/argument analyser and a modified version of the parser described in the previous section. The changes to the parser will involve creating a new data structure, adding new rules to the grammar, and allowing the grammar interpreter limited additional power. However, we will not sacrifice the fundamental characteristic of the parser which is that it operates strictly deterministicly with limited access to contextual information.

3.2.1 A First Attempt

The conjunction condition proposed in Chapter 2 can be restated as follows: for two constituents to be conjoined in a sentence, each constituent must be able to syntactically and semantically (with respect to predicate/argument relationships) replace the other in the position in the sentence held by their conjunction. Given that the parser is not to be fundamentally altered, it will in some cases be necessary for it to conjoin two constituents before these two conditions can be verified, and without having the power to destroy the conjunction should it be subsequently determined that the conditions are not satisfied. That is, the most the parser can do (and this seems to be sufficient) is to ensure that, given the information associated with the nodes to which it has access, it is not impossible (1) for the two constituents to syntactically replace each other (i.e., given their grammatical features they could fill the same position in the tree), and (2) for the two constituents to have the same predicate/argument relationships (i.e. given their semantic features they could fill the same predicate/argument frame slots). The sufficiency of this follows from Marcus's determinism hypothesis, upon which the design of his parser was based.

Of course, the rules to handle conjunction must be written and their priorities set in such a way as to allow the parser to examine the relevant context before taking any irreversible action. We first describe a simple approach and show its insufficiency.

Suppose we were to add to the grammar a set of rules, each having a pattern of this general form

[=<node-type>][=*and][=<node-type>]

and an action specifying to conjoin the two nodes in the buffer satisfying the description [=<node-type>]. These rules would correctly parse a number of conjunctions. However, it is possible for two conjoined constituents to contain subconstituents whose dominating nodes also satisfy the pattern of one of these conjunction rules. This is illustrated by sentence (3) below.

(3) Jack likes Mary and Peter likes Susan.

The constituents *Jack likes Mary* and *Peter likes Susan* which are conjoined in this sentence contain the subconstituents *Mary* and *Peter* whose dominating nodes would satisfy the pattern of the rule to conjoin two NP nodes.

Simply giving these conjunction rules low priorities does not solve the problem. For example, the rule to conjoin two NPs might be given a priority lower than that of the rule which attaches an NP to a VP node as an object, so that sentence (3) would be correctly parsed. However, there are sentences, like (4) below, in which two objects are conjoined.

(4) Jack likes Mary and Peter.

If the NP conjunction rule had a priority lower than the rule attaching an NP as an object, the parser would attach the NP *Mary*, leaving the conjunction and the NP *Peter* stranded in the buffer. One solution to this problem is to augment the patterns of the simple conjunction rules to test for appropriate contexts. Given the above two examples, it appears that the pattern for the rule which conjoins two NPs should be

[=np][=*and][=np][not =verb].

However, this pattern is appropriate only if the parser is looking for an object NP. It would not be satisfied by two conjoined subject NPs, as in (5) and (6) below.

(5) Jack and Mary like Peter.
(6) Susan believes that Jack and Peter like Mary.

Thus, two other rules with the patterns

[=np][=*and][=np][=verb] and [=comp,*that][=np][=*and][=np][=verb]

are necessary. These three NP conjunction rules would have to be in different packets so that they would be applied only at appropriate points in the parse.

3.2.2 The Method

Using the approach described in the previous section, a new conjunction rule is needed for each context in which a node of a given type can appear. Unfortunately, this results in a large grammar requiring a complicated priority scheme. Furthermore, the approach cannot in general handle an arbitrary number of conjoined constituents. Therefore, this approach to the problem will not be used, but rather another which is simpler and seems to more accurately reflect the way in which speakers of English analyse sentences of their language.

This approach initially involves adding to the grammar a general conjunction rule CONJOIN-NODES (figure 3) to perform the functions of the the simple low-priority conjunction rules proposed in the previous section.

{AS RULE CONJOIN-NODES PRIORITY: low IN EVERYWHERE
[* is compatible with 3rd][=*and][t] -->
Create a new node of type 1st.
%"type" returns the type of a node.%
Attach 1st to c as type 1st.
Attach a new conjp node to c as conjp.
Attach 2nd to c as conj.
Attach 3rd to c as type 3rd.
Drop c %the conjp node%.
Drop c %the new type 1st node%.}

Figure 3. Grammar Rule CONJOIN-NODES

The priority of CONJOIN-NODES is set sufficiently low to ensure that it is executed only if no other rules could apply. The first description in the pattern of this rule

[* is compatible with 3rd]

is a predicate/argument test which determines whether or not it is possible for the two constituents in 1st and 3rd to fill some common predicate/argument slot.

Given this first step two problems remain to be solved. One is the problem described above in which the left conjunct is prematurely attached leaving the conjunction in the first buffer cell. The other is that the pattern of CONJOIN-NODES will not be satisfied if the left conjunct is in the second or third buffer cell. (Actually, if the left conjunct is in the third buffer cell there will always be a rule with a priority higher than that of the conjunction rule whose application will result in the node being moved up in the buffer.)

There are two solutions to the latter problem. One is to have two general conjunction rules, one having the pattern

[* is compatible with 3rd][=*and][t],

to handle the case where the left conjunct is in C(1), and the other having the pattern

[t][* is compatible with 3rd][=*and][t],

to handle the case where the left conjunct is in C(2). This solution is unappealing because: (1) the buffer window would have to be extended from three to four cells to accommodate the second pattern; and (2) it fails to capture a possible generalization as both rules would have only minimally different patterns and actions.

A better solution is to modify the attention shifting (AS) mechanism so that an AS rule can have a low priority. (Currently, attention shifting is implemented in such a way that AS rules implicitly have high priorities.) The rule CONJOIN-NODES could then be a low priority AS rule and its pattern tested starting at any cell in the buffer window.

The problem of attaching the left conjunct before the right conjunct has been parsed will be solved in the following way. Rather than ensuring that the parser never prematurely attaches the left conjunct, new rules will be added to the grammar to take the appropriate action in case it does. This will necessitate giving additional powers to the grammar interpreter; for instance, the power to detach a single node from its position in the tree so that two conjuncts can be conjoined.

Before presenting the new grammar rules and interpreter powers, we will describe the motivation behind them. Suppose again that the parser is given sentence (2) (repeated below as (5)) and the parse has progressed to the state depicted in figure 4.

(5) Jack likes Susan and Peter likes Mary.

Rule about to run: OBJECTS in SS-VP

Active Node Stack:

ROOT (<features>) / (<associated packets>)

S1 (...) / (...)
NP: (Jack)

c: VP1 (...) / (CPOOL INF-COMP SS-VP)
VERB: (likes)

Buffer:

1: NP2 (...): Susan

Figure 4. Parsing the VP *likes Susan.*

At this point the rule with the highest priority whose pattern is satisfied is the rule OBJECTS ([=np]). As a result of its application the NP in C(1) is attached to **c**. The conjunction is now in C(1) and the pattern of the rule VP-DONE ([t]) matches the buffer. Its application causes **c** to be dropped and S1 becomes the new current active node. Later it will be necessary to assume that whenever a constituent is complete, as S1 is in this example, there is a low priority rule which drops it into the buffer. The tree structure so far is shown in figure 5.

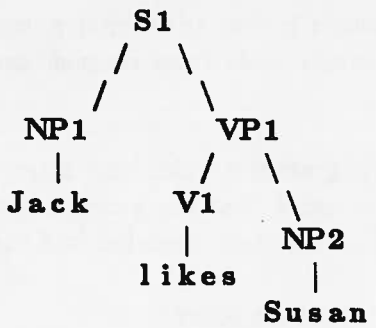


Figure 5. Tree representation of *Jack likes Susan.*

The parser cannot determine, without examining the input following the conjunction, which constituent preceding the conjunction is the left conjunct. However, it is possible to determine a subset of the current set of nodes which excludes those nodes dominating constituents which could not possibly be the left conjunct. This is based on the observation that the left conjunct must contain the word which immediately precedes the conjunction. In other words, only those nodes which are rightmost (i.e., have no brothers to their right) in the parse tree of the sentence up to the conjunction are potential left conjuncts. For a given conjunction, such a node will be referred to as an RM (RightMost) node associated with the conjunction and the set of such nodes as the RM set associated with the conjunction. In the above example, the RM set associated with the conjunction in 1st consists of the nodes S1, VP1, and NP2.

Given that the parser has the facility to determine the set of RM nodes associated with a conjunction, it still must determine which of the nodes in the RM set is the left conjunct. This cannot be achieved without examining the input following the conjunction. To do this, the parser must know what it is looking for so that it may activate the relevant rule packets. The coordinate conjunction condition states that the right conjunct must be able to syntactically replace the left conjunct. Therefore, the parser must activate all those packets containing the rules to parse any constituent which could replace one of the RM nodes. For example, if the RM set contains the nodes from the above example, RM={S1, VP1, NP2}, the parser must be prepared (i.e., have activated the relevant packets) to parse any constituent which could replace S1 in its context; VP1, in its context; or NP2, in its context.

The packets that are active at the time a node A is attached contain all the rules to parse and attach syntactic replacements for A; therefore the parser must be able to recall the packets that were active when each of the RM nodes was attached. For example, consider again the parse of sentence (5). When NP2 (*Susan*) is attached to VP1, the packets CPOOL, SS-VP, and INF-COMP are active. The following are examples of syntactic replacements for NP2 and the rules to initiate their parses:

Jane (Jack likes Jane.): STARTNP in CPOOL

to sing (Jack likes to sing.): CREATE-TRACE-SUBJECT in INF-COMP

Susan to sing (Jack likes Susan to sing.): INF-S-START-1 in INF-COMP

In our grammar, all these constituents are eventually dominated by an NP node, which could be attached to VP1 by the same rule which attached NP2 (*Susan*) to VP1, namely, OBJECTS in SS-VP. Note that the NP *Jane* could be conjoined to NP2 since it is able to fulfill the predicate/argument role fulfilled by *Susan*, namely neutral of the verb *likes* with the sense which takes a physical object as its neutral case. The constituents *to sing* in *Jack likes to sing* and *Susan to sing* in *Jack likes Susan to sing* do not fulfill this predicate/argument role. Both function as the neutral of the verb *likes* with the sense which takes a clause as its neutral case. Thus, sentence (7) below is grammatical and sentences (8) and (9) are not.

(7) Jack likes Susan and Jane.
(8) *Jack likes Susan and Susan to sing.
(9) *Jack likes Susan and to sing.

Given that the parser can determine which nodes are the RM nodes associated with a conjunction and can also recall the packets that were active when each RM node was attached, it can activate these packets to parse the right conjunct. However, some of these packets would not normally be concurrently active and thus the relevant priorities would not have been assigned to the rules they contain. For example, SS-VP contains the rule OBJECTS whose pattern is [=np] and whose priority is 10 and SS-START contains the rule MAJOR-DECL-S whose pattern is [=np][=verb] and whose priority is also 10. SS-VP and SS-START would both be activated for the parse of the right conjunct of the sentence, *Jack likes Susan and Peter likes Mary.* Given the current rule priorities, the NP *Peter* could be incorrectly chosen as the conjunct of *Susan.* This priority problem will be discussed in detail in section 3.2.3.

3.2.3 The Grammar Rules for Parsing Conjunctions

We now introduce the new grammar rules to carry out the tasks which have just been described. The first rule is NOTE-RM-NODE which "notes" an RM node as soon as it is completed, i.e. as soon as it is dropped from the stack into the buffer. Recall that an RM node associated with a particular conjunction is one that dominates a constituent immediately preceding the conjunction. As long as there is no reordering of the input (i.e., the constituent in C(i) is al-

ways attached to the current active node before the constituent in C(j), for i<j)[1] any node which at some point immediately precedes the conjunction in the buffer is an RM node. Since completed nodes are always dropped into the first buffer cell, the following pattern is sufficient to recognize an RM node which has not yet been noted:[2]

[* is not RM-noted][* is *and].

There are several ways for the parser to note an RM node. For example, it could simply label the node with an appropriate feature. However, so that the parser may later have easy access to these nodes, the node will be made a member of a set associated with the conjunction called the conjunction's "RM set". Note that each conjunction occurring in a sentence will have its own RM set and that it is not possible for a node to be in more than one RM set.

The following new action command will be added to the PIDGIN grammar language to note a node which is an RM node with respect to another node:

Note <node> as an RM node associated with <node>.

NOTE-RM-NODE must have a high priority so that a node is noted before it is attached. It must also be contained in all packets since nodes of almost every type can appear in conjunctions. The rule NOTE-RM-NODE is given in figure 6 below.

{RULE NOTE-RM-NODE PRIORITY:high IN EVERYWHERE
[* is not RM-noted][* is *and] -->
Note 1st as an RM node associated with 2nd.
Label 1st RM-noted.}

Figure 6. Grammar Rule NOTE-RM-NODE

To ensure that any constituent which could be the left conjunct gets noted, it is necessary for a node to be dropped into the buffer before it is attached to the node above it in the active node stack. Thus, the PIDGIN command, "attach a new <type> node to c as <type>", which attaches the new node immediately after its creation, should not be used if the new node

---

1. This is not true of Marcus's [1980] grammar (eg. AUX-INVERSION), but it is of the grammar given in Appendix A.

2. There is a problem with nodes created by attention shifting rules since the buffer window is slid back before the pattern of NOTE-RM-NODE is tested. A solution is to have the parser note an RM node just before it is dropped; i.e., as a consequence of the "Drop c" command. The NOTE-RM-NODE rule would still be required to note words (which are never c) which immediately precede a conjunction.

could be a left conjunct. If the new node is attached, it will not be inserted into the buffer when it is dropped and thus will not be noted.[3] It is also necessary for there to be low priority rules which pop the current active node from the stack if it dominates a complete constituent. These rules must have low priorities relative to other rules which could be active at the same time to avoid prematurely popping c from the stack. Our grammar contains such rules; for example, VP-DONE in VP-FINAL and S-DONE in SS-FINAL which both have priority 20.

Whenever an RM node is attached to c, the parser will store the packets which are associated with c so that they may be later activated to parse syntactic replacements for the node. An RM node may thus have two sets of associated packets: the packets which were active when the node was attached to c, and the packets which were active the last time the node was itself c. To avoid confusion, we will refer to the packets which were active when the node was attached to c as the node's R-packets, where the R stands for "replacement". (A node's R-packets are the packets to parse syntactic replacements for the node.) The packets which were active the last time the node was itself c will be referred to as the node's C-packets.

The next new conjunction rule is PARSE-CONJP which starts the parse of the conjunctive phrase (figure 7). Like NOTE-RM-NODE, this rule is contained in every packet and so is always active. The pattern of the rule is simply [=*and]. The body of the rule specifies to attach 1st (*and) to a new CONJP node and to activate the R-packets of the nodes in the conjunction's RM set. It also activates another packet, CONJ-FINAL, which is described below. PARSE-CONJP must have a priority lower than all other rules so that it is not run until the parse of the preceding constituents has progressed as far as possible.

{RULE PARSE-CONJP PRIORITY:25 IN EVERYWHERE
[=*and] -->
Create a new conjp node.
Attach 1st to c as conj.
Activate conj-final, R-packets of the RM-nodes associated with 1st.}

Figure 7. Grammar Rule PARSE-CONJP

The attachment of the conjunction to a new CONJP node in the active node stack and the activation (with respect to this node) of the R-packets associated with the RM nodes serves three purposes:

1) Because the conjunction is removed from C(1), all the buffer cells are free for the parse of the right conjunct.
2) The CONJP node in the active node stack serves as a separator between the parse of the

3. If the parser notes an RM node immediately before dropping it, as suggested in footnote (2), this is not a problem.

right conjunct and the part of the parse of the sentence preceding the conjunction.

3) If the CONJP node is **c** and the node in C(1) can syntactically replace an RM node A, then there is a rule in one of A's R-packets (and therefore active) whose pattern will match the buffer and whose action will specify the attachment of the node in C(1) to **c**. If the CONJP node is **c** and the node in C(1) is potentially a subnode of a node which can syntactically replace an RM node A, then there is a rule in one of A's R-packets whose pattern will match the buffer and whose action will specify to create the node. If a new node is created, the CONJP node will be pushed back into the stack and it's C-packets (i.e., CONJ-FINAL and the R-packets of the RM nodes) will become inactive until the new node is popped off the stack.

Note that if a node N is attached to the CONJP node by a rule in packet P, then N is a syntactic replacement for all and only those RM nodes whose R-packets contain P. Therefore P, called the "Syntactic Replacement" (SR) packet, will be stored by the parser when N is attached, so that it may later be used to recall the RM nodes which N can syntactically replace.

As mentioned above, another packet, CONJ-FINAL, is activated by the rule PARSE-CONJP. This packet contains one rule, CHECK-RIGHT-CONJ, whose action is to drop the CONJP node once a constituent has been attached to the right of the CONJ node and then to determine if this node is the right conjunct. This rule will be described in detail later on in this section.

Once all these packets have been activated, the parser must decide which rule to test first. Ordinarily, it tests the rules in order of descending priority and runs the first whose pattern matches the buffer. We mentioned above that new relative priorities must be assigned to some of the rules contained in packets which are not normally co-active. The rule pairs which are affected are those whose patterns could both potentially match the buffer. For example, the relative priorities of the two rules VERB ([=verb]) and MAJOR-DECL-S ([=np][=verb]) need not be changed since their patterns could never both match the buffer.

A first attempt at determining the appropriate relative priorities might be to apply Marcus's "further specification" rule. This rule states that if the pattern of a rule R_1 is a further specification of the pattern of another rule R_2 (e.g., [=np][=verb] is a further specification of [=np]), then R_1 will have a higher priority than R_2. However, not only does this rule not always apply (since two patterns can match the buffer without one being a further specification of the other), but it does not always give the desired result when it does apply. Consider, for example, sentence (8) below.

(8) Jack has the cups and saucers for the coffee.

The parse tree of the sentence up to the conjunction is shown is figure 8.

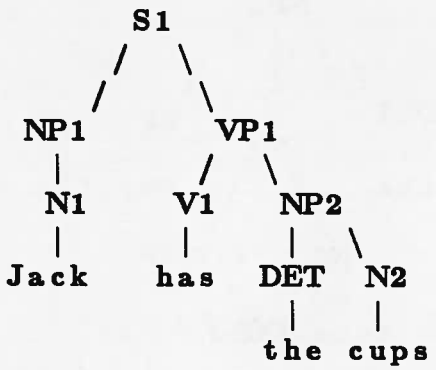


Figure 8. Parse tree for *Jack has the cups.*

The RM set consists of the nodes S1, VP1, NP2, and N2. Once the CONJP node has been created and the packets activated, the parser must choose to apply one of the following two rules whose patterns will match the buffer: NOUN ([=noun]) in PARSE-NOUN which is an R-packet of N2 and OBJECTS ([=np]) in SS-VP which is an R-packet of NP2. If the rule OBJECTS is chosen, the NP *saucers for the coffee* will be attached to the CONJP node as the right conjunct and then conjoined to NP2.[4] If the rule NOUN is chosen, the noun *saucers* will be attached to the CONJP node as the right conjunct and then conjoined to N2. The PP *for the coffee* will later be attached to NP2 as a modifier of the conjoined nouns. This is the desired parse. It is illustrated in figure 9.

4. The pattern of OBJECTS ([=np]) matches the buffer only after *saucers for the coffee* has been parsed as an NP. The parse of this constituent, started by the AS rule START-NP in CPOOL (an R-packet of NP2), only temporarily interrupts the matching of OBJECTS' pattern.

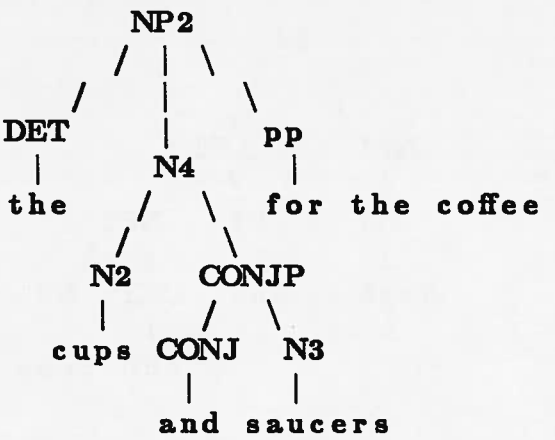


Figure 9. Desired parse for *the cups and the saucers.*

It would appear that given two rules in packets associated with different RM nodes, the rule in the packet associated with the lower RM node should have the higher priority. This will be called the "lowest node" priority rule. As further evidence for this rule, consider sentences (10)-(14) below, which were presented to a number of people.

(10) Jack has the cups and saucers are on the table.

(11) Jack has cups for the coffee and tea will be served in glasses.

All were "led down the garden path" on both of the above sentences by conjoining *cups* and *saucers* in (10) and *coffee* and *tea* in (11). That is, the nouns following the conjunction were conjoined to the nouns immediately preceding the conjunction WITHOUT checking to see if they could belong to larger constituents. However, no one seemed to be confused by the following sentence.

(12) Jack has the cups and Jill has the saucers.

The left conjunct in the above sentence is identical to that in (11); the right conjunct begins with the proper noun *Jill.* One might argue that in (12) *Jill* is initially taken to be the right conjunct, as is *saucers* in (11), but is rejected because of its predicate/argument incompatibility with *cups.* However, this argument would suggest that (13) below is a garden path sentence. This does not appear to be the case.

(13) Jack likes Susan and Peter likes Mary.

To further confuse the issue, many people found the following sentence mildly troublesome.

(14) The text of the book was written by Chris and Kathy did the illustrations.

The above phenomena might be explained in the following way. Once a noun or NP following the conjunction has been parsed, not only is it compared with the possible left conjuncts for predicate/argument compatibility but also with the subject NP of the sentence. If the constituent is predicate/argument compatible with the subject as well as one of the RM nodes, the reader becomes aware of the possibility that it is the subject of a new sentence, and so does not immediately conjoin it to the RM node. This would explain why no one had trouble with sentence (13). If the node is compatible with an RM node but not with the subject, then there is no reason for the reader to consider the possibility that it is a subject, and so it is conjoined to the RM node. This would explain why everyone found (10), (11), and (14) confusing, to greater or lesser degrees.

There appear to be other factors affecting the degree to which sentences like the ones above are confusing to a reader. For example, an NP following a conjunction whose head noun has has the semantic feature *animate* is more readily taken to be a subject than a NP whose head noun does not have this feature. The fact that people are much less confused by (14) than they are by (10) and (11) is evidence of this.

Given a priority scheme based either on the further specification rule or the lowest node rule, it is possible that executing the rule with the higher priority will cause the parser to attach a constituent which is not the right-hand conjunct to the CONJP node. However, if the parser uses the lowest node rule the power necessary to continue the parse can be limited, given that the sentence is not a garden path. Using this rule, the smallest constituent following the conjunction which can syntactically replace an RM node is attached to the CONJP node, even if it could belong to a larger constituent. If the node is determined not to be the right conjunct then the parser will need the power to detach it from the CONJP node and return it to the buffer but will never have to destroy existing nodes. The parser must have the power to detach a node in any case, so that it can detach the left conjunct before performing the conjunction. It is important to note that giving the parser this power does not allow it to do general backtracking. Any node that is detached by the parser must be dropped into the buffer; thus no nodes are ever destroyed. Furthermore, any node that is detached must reappear intact somewhere else in the tree. For example a node which is detached from a CONJP node will be dominated by the CONJP node in the final parse tree, although not necessarily directly dominated by it.

The lowest node priority scheme also provides a way to distinguish between major garden path conjunctions and minor ones. Consider, for example, sentence (15) below.

(15) Jack brought wine and juice for those who don't drink alcohol.

Sentence (15) is only mildly confusing because, although the reader conjoins *wine* and *juice* before examining the following prepositional phrase *for those who don't drink alcohol* (which cannot modify *wine*), it is fairly easy for the reader to conclude that the PP modifies only *juice* and

that this larger constituent forms the right conjunct.

If the lowest node priority scheme is used, such garden path sentences can be resolved without destroying existing nodes, in particular the node dominating the CONJP node need not be destroyed; the node mistaken for the right conjunct need only be detached and the larger constituent attached in its place. To resolve major garden path sentences like (10) and (11) it is necessary to destroy at least the node dominating the CONJP node.

There are two more rules to be introduced: CHECK-RIGHT-CONJ and CONNECT-NODE-AND-CONJP. CHECK-RIGHT-CONJ is a high-priority rule in CONJ-FINAL and so is active whenever the CONJP node is c. Its pattern is satisfied when a potential right conjunct has been attached to the CONJP node (i.e., when a node has been attached to the CONJP node to the right of the CONJ node). The action taken by the rule is determined as follows. The potential right conjunct, R, is compared for predicate/argument compatibility to the RM nodes which R can syntactically replace (i.e., the RM nodes whose R-packets contain the SR packet).[5] If R is an N or an NP and there is an S node in the RM-set, then R is also compared to the subject of the S node (or the lowest S node if there is more than one). If R is not compatible with any of the relevant RM nodes then the priority of the rule which attached R is temporarily lowered and R is put back in the buffer. Otherwise, if R is compatible with the subject of an S node, then R is appropriately labelled, put back into the buffer, and the R-packets associated with the S node are temporarily given higher priority so that if there is a new sentence it is parsed immediately.

If R is compatible with one of the RM nodes and it is not compatible with the subject of an S node in the RM-set, then R is taken to be the right conjunct. If R is compatible with only one RM node then that node is the left conjunct; otherwise comparative semantic tests of the sort described in section 3.1.2 will have to be performed to determine which of the RM nodes is most likely to be the left conjunct. Once the left conjunct has been determined, the CONJP node is dropped into the buffer. The left conjunct is then detached from its dominating node and dropped into the buffer. Since the conjunction must be attached to the node to which the left conjunct was previously attached, this node is made the current active node.

The high-priority rule CONNECT-NODE-AND-CONJP performs the conjunction. This rule is contained in every packet and hence is always active. Its pattern is [t][=conjp] and its action creates a new node whose type is the same as that of the left conjunct (which is in 1st), attaches the left conjunct and the CONJP node to the new node, and then drops it into the buffer. This new node will be attached to the node which previously dominated the left conjunct (and is now the current active node) by the same rule which originally attached the left conjunct in its place. The rule CONNECT-NODE-AND-CONJP is displayed in figure 10.

---

5. Recall that a node N_1 is predicate/argument compatible with another node N_2 if for each frame in which N_2 appears, N_1 can fill the slot filled by N_2 in at least one of the hypotheses for the frame.

{RULE CONNECT-NODE-AND-CONJP PRIORITY:5 IN CONJ-FINAL
[t][=conjp] -->
Label a new node type 1st.
%"type" is a function which returns a node's type.%
Attach 1st to c as type 1st.
Attach 2nd to c as conjp.
Drop c.}

Figure 10. Grammar Rule CONNECT-NODE-AND-CONJP

3.2.4 An Example

We conclude with an example illustrating the priority scheme, the new rules, and the additional parser powers we have just described.

Suppose the parser is given sentence (14) as input and the parse has progressed to the state depicted in figure 11.

(14) Jack likes Mary to sing and to play the piano.

Rule about to run: NOTE-RM-NODE in EVERYWHERE

Active Node Stack:

ROOT (...)/(...)

S1 (...)/(...)
NP: (Jack)

VP1 (...)/(...)
WORD2: (likes)

c: S2 (INF-S ..)/(PARSE-VPS SS-FINAL)
NP: (Mary)

Buffer:

1: VP2 (aux,to ..): (to sing)
2: WORD6 (*and ..): (and)

Figure 11. Parsing the infinitive phrase *Mary to sing*

VP2 is about to be noted as an RM node. After this is done it is attached to S2 by the rule ATTACH-TOP-VP and the active packets, PARSE-VPS and SS-FINAL, are stored as VP2's R-packets. S2 is then dropped, noted as an RM node, and attached to a new NP node, NP3. The packets active at this point are stored as S2's R-packets. NP3 is then dropped, noted, and attached to VP1; VP1 is dropped, noted, and attached to S1; and S1 is dropped, noted, and attached to the ROOT node.

At this point, the conjunction is in the first buffer cell and the pattern of the rule PARSE-CONJP is satisfied. The rule action creates a CONJP node, attaches the conjunction to it, and activates CONJ-FINAL and the R-packets associated with the RM nodes. The state of the parser just after these packets have been activated is depicted in figure 12 below.

Active Node Stack:

ROOT (...)/(...)
S: (Jack likes Mary to sing)

c: CONJP1 (...)/(CONJ-FINAL PARSE-VPS TO ..)
WORD6: (and)

Buffer:

1: WORD7 (*to verb ..): (to)
2: WORD8 (*play verb mainverb ..): (play)

RM nodes associated with WORD6:

V3: sing (PARSE-VERB)
VP3: sing (PARSE-VPS TO)
VP2: to sing (PARSE-VPS TOP-VP)
NP3: Mary to sing (CPOOL, SS-VP)
S2: Mary to sing (INF-COMP COMP-S-FINAL)
VP1: likes Mary to sing (PARSE-VPS TOP-VP)
S1: Jack likes Mary to sing (SS-START CPOOL)

Figure 12. Parsing the conjunctive phrase.

The packets PARSE-VPS (an R-packet of VP1, VP2, and VP3) and INF-COMP (an R-packet of S2) are among those activated with respect to the CONJP node to start the parse of the right conjunct. There is a rule in each of these packets, START-VP in PARSE-VPS and CREATE-TRACE-SUBJECT in INF-COMP, whose pattern matches the input following the conjunction. The pattern of START-VP is [=verb] and the pattern of CREATE-TRACE-SUBJECT is [=*to,auxverb][=tnsless]. Both patterns match the buffer, which contains the word *to* in 1st and the word *play* in 2nd. Although the constituent *to play the piano* is a syntactic replacement for both VP2 *to sing* and S2 *Mary to sing* as shown in (15a-b) and (16a-b), VP2 is preferred as the left conjunct to the extent that the reading in which S2 is the left conjunct might not even be perceived by a speaker of English.

(15a) Jack likes [[Mary] [to sing]].
(15b) Jack likes [[Mary] [to play the piano]].

(16a) Jack likes [[Mary] [to sing]].

(16b) Jack likes [[t:Jack] [to sing]].

Because PARSE-VPS is associated with the lower node in the RM set (VP2), the rule START-VP has priority and the string *to play the piano* is parsed as a VP. Its structure is given in figure 13 below.

```
VP4(to, aux)
  WORD7(verb, ..)  to
  VP5(main)
    WORD8(verb, main, ..)  play
    NP3
      WORD9(det, ..)  the
      WORD10(noun, ..)  piano
```

Figure 13. The parser output for *to play the piano.*

Once VP4 has been parsed it is dropped into the buffer. It is then attached to the CONJP node by the rule ATTACH-TOP-VP which is in the packet TOP-VP. The packet TOP-VP is stored as the SR packet. Since the CONJP node now has two sons, the rule CHECK-RIGHT-CONJ applies. The rule determines that VP4 is predicate/argument compatible with the node VP2, the only RM node for which TOP-VP is an R-packet, so it drops the CONJP node, detaches VP2 from S2, and makes S2 the current active node by pushing it onto the stack. The state of the parser at this point is shown in figure 14.

Rule about to run: CONNECT-NODE-AND-CONJP in EVERYWHERE

Active Node Stack:

ROOT (...)/(...)
S: (Jack likes Mary..)

c: S2 (...)/(...)
NP: (Mary)

Buffer:

1: VP2 (...): (to sing)
2: CONJP1 (...): (and to play the piano)

Figure 14. Connecting the left conjunct and CONJP.

CONNECT-NODE-AND-CONJP creates a new VP node (VP5), attaches VP2 and CONJP1 to VP5, and then drops VP5 into the buffer. There the node satisfies the pattern of the rule ATTACH-TOP-VP in TOP-VP which attaches it to S2. S2 is then dropped and, since it is already attached to VP1, is not inserted into the buffer. The node which was above S2 in the stack (ROOT) becomes the current active node. Figure 15 depicts the final state of the parser for this example.

Rule about to run: FINAL-PUNC in ROOT-FINAL

Active Node Stack:

c: ROOT (...)/(ROOT-FINAL)
S: (Jack likes Mary to sing and to play the piano)

Buffer:

1: WORD11 (*. FINAL-PUNC ..): (.)

Figure 15. Parsing the final punctuation.

3.3 Summary

In this chapter we described a method of parsing coordinate constituent conjunctions which uses the CCC condition proposed in chapter 2. The parsing system, which was modelled after the PARSIFAL system developed by Marcus, includes two main components: a parser and a predicate/argument analyser. It was necessary to make three major modifications to the parser:

1) A new power: The parser was given the power to detach a single node from the tree and insert it into the buffer. This does not allow general backtracking since nodes cannot be destroyed. A node which is detached must appear intact in the final tree structure.

2) A new priority scheme: The static priority scheme used by Marcus was determined to be insufficient for parsing conjunctions and a dynamic relative priority rule was proposed as a supplement to it. The rule, called the "lowest node" priority rule, is used to determine the relative priorities of two grammar rules which are in R-packets associated with RM nodes. It states that the rule associated with the lower RM node has the higher priority.

3) RM sets and R-packets: Each node was allowed two more associated sets -- an RM set and a set of R-packets. The RM set associated with a node N contains the nodes which are rightmost in the tree representing the part of the sentence immediately preceding the constituent dominated by N. If N is a conjunction then its RM set contains all the possible left conjuncts for N. The R-packets associated with a node N contain all the rules necessary to parse syntactic replacements for N in its grammatical environment.

The following new rules were added to the grammar to parse coordinate conjunctions:

The rule CONJOIN-NODES will apply if no higher priority rule applies to attach the left conjunct. For example, it would conjoin the NPs *Peter* and *Paul* in: *I believe that Paul and Peter have left.*

```
{AS RULE CONJOIN-NODES PRIORITY: low IN EVERYWHERE
[* is compatible with 3rd][==*and][t] -->
Create a new node of type 1st.
%type returns the type of a node.%
Attach 1st to c as type 1st.
Attach a new conjp node to c as conjp.
Attach 2nd to c as conj.
Attach 3rd to c as type 3rd.
Drop c %the conjp node%.
Drop c %the new type 1st node%.}
```

The following rules handle the case in which the left conjunct is attached before the right conjunct is parsed. All of these rules would be used to parse the conjunction in: *I believe Paul and Peter.*

{RULE NOTE-RM-NODE PRIORITY: high IN EVERYWHERE
[* is not RM-noted][* is *and] -->
Note 1st as an RM node associated with 2nd.
Label 1st RM-noted.}

{RULE PARSE-CONJP PRIORITY: low IN EVERYWHERE
[==*and] -->
Create a new conjp node.
Attach 1st to c as conj.
Activate conj-final, packets in RM set associated with 1st.}

{RULE CHECK-RIGHT-CONJ PRIORITY: high IN CONJ-FINAL
[**c; * has two sons] -->
%The action of this rule will try to find an RM node
whose p/a function can be fulfilled by c's second
son (R).
If there is not a compatible RM node then R is
detached from c and the rule which attached it is
removed from the set of active rules.
If there is a compatible RM node and R can fulfill
the p/a function of the subject of an S node in the
RM-set then R is labelled, detached from c, put back
into the buffer and the priorities of the rules in the
R-packets associated with the S node are temporarily
raised.
If there is a compatible RM node and R cannot fulfill
the p/a function of the subject of an S node in the
RM-set then the son is taken to be the right conjunct
and the RM node, the left conjunct.
C (the conjp node) is dropped into the buffer and then
the left conjunct is detached from its parent, which
is made the new c, and the left conjunct is dropped
into the buffer.%}

{RULE CONNECT-NODE-AND-CONJP PRIORITY: high IN EVERYWHERE
[t][=conjp] -->
Create a new node of type 1st.
Attach 1st to c as type 1st.
Attach 2nd to c as conjp.
Drop c.}

4. Conclusions and Future Work

This thesis investigated the use of the English word *and* as a coordinate conjunction of complete constituents. It proposed a necessary and sufficient condition for the acceptability of an English sentence containing a coordinate constituent conjunction, which was composed of two general constraints — one referring to syntactic relationships and the other, to predicate/argument relationships. While it was not possible to prove the validity of the condition (since there is no complete formal grammar of English), the condition has a sound intuitive basis and was demonstrated to correctly predict the acceptability of a large number of sentences containing coordinate conjunctions.

The constraints were incorporated into a deterministic parsing system modelled after the PARSIFAL system developed by Marcus [1980]. Although a number of modifications were made to his original system, the fundamental characteristic of the parser, which is that it operates strictly deterministicly with access to limited contextual information, was not sacrificed.

Although the thesis discussed only the coordinate conjunction of complete constituents, it appears that it is possible to generalize the results to handle conjunctions of incomplete constituents in which the conjuncts "share" one or more constituents. The general form of a sentence containing such a conjunction is:

$$S = X - [[C_1 \text{ and } C_2]\ Y] - Z,$$

where Y is the string of constituents which are shared by C_1 and C_2; i.e., the constituents of Y are logically subconstituents of both C_1 and C_2. Three examples of this general structure are given below.

(1) Chris whistled, and Mary sang, a lively tune.

(2) The man who won and the man who lost the leadership race last weekend are both good politicians.

(3) Peter was shown, and is now convinced, that smoking is unhealthy.

The necessary and sufficient condition for coordinate constituent conjunction (CCC) proposed in the thesis can be altered slightly to provide a necessary, although not sufficient, condition for the coordinate conjunction of incomplete constituents (CCIC).

(C4) A Necessary Condition for CCIC

The sentence $S = X - [[C_1 \text{ and } C_2]\ Y] - Z$ is grammatical ONLY IF for

$$S_1 = X - C_1' - Z \text{ and } S_2 = X - C_2' - Z,$$

where for i=1,2: if [C_i Y] is an NP then C_i' is the plural form of [C_i Y], otherwise C_i' is [C_i Y],

both (i) and (ii) hold:

(i) S_1 and S_2 have acceptable syntactic analyses $A(S_1)$ and $A(S_2)$ which are identical except that the position filled by C_1' in $A(S_1)$ is filled by C_2' in $A(S_2)$.

(ii) S_1 and S_2 have acceptable predicate/argument analyses $P(S_1)$ and $P(S_2)$ which are identical ignoring C_1', C_2', and their subconstituents; and the roles that are filled by C_1' in $P(S_1)$ are filled by C_2' in $P(S_2)$.

Given the analyses below, it is easy to verify that sentences (1)-(3) satisfy the condition.

(1) Chris whistled, and Mary sung, a lively tune.

[$_{S_1=C_1'}$ Chris whistled a lively tune]

[$_{S_2=C_2'}$ Mary sung a lively tune]

(2) The man who won, and the man who lost, the leadership race last weekend are both good politicians.

[$_{S_1}$ [$_{C_1'}$ The men who won the leadership race last weekend] [$_{VP}$ are both good politicians]]

[$_{S_2}$ [$_{C_2'}$ The men who lost the leadership race last weekend] [$_{VP}$ are both good politicians]]

(3) Peter was shown, and is now convinced, that smoking is unhealthy.

[$_{S_1}$ [$_{NP}$ Peter] [$_{C_1'}$ was shown that smoking is unhealthy]]

[$_{S_2}$ [$_{NP}$ Peter] [$_{C_2'}$ is now convinced that smoking is unhealthy]]

An examination of many other sentences containing such structures has indicated that the above condition is not sufficient; constraints on the shared constituents are also necessary. For instance, it appears that no more than one NP or complement S constituent may be shared, and that a shared complement S must be finite. These two restrictions are demonstrated by the following sentences which satisfy (C4), but which are ungrammatical (or at best questionable).

Two NPs shared:

(4) *I gave, and Peter loaned, Jack some textbooks.

An NP and a complement S shared:

(5) *Gord told, and Nancy convinced, Ted that smoking was unhealthy.

An infinitive complement S shared:

(6) *?Betty wants, and Joyce expects, Bill to pay for dinner.

Otherwise, contrary to Gazdar [1981], there does not appear to be a general constraint on the number of shared constituents. One can easily generate a number of grammatical sentences in which two and three constituents are shared by the conjuncts. For example, replacing the NP NP sequence in sentence (4) with an NP PP sequence results in an acceptable sentence.

(7) I gave, and Peter loaned, some textbooks to Jack.

There also appears to be no constraint on the depth of the node which logically dominates a shared constituent (i.e., the node to which the shared constituent or its trace would be attached). In the following sentence the node in the right conjunct which logically dominates the shared constituent is in a deeply nested VP.

(9) I have been looking for, and Peter said that Mary told Joe that she was unable to find, the course textbook.

Of course, human memory limitations can affect the ability to understand a sentence in which a large amount of material intervenes between the left conjunct and the first shared constituent.

While the constraints on CCIC are straightforward, their incorporation into a parser like the one described in the thesis is not. Recognizing constituents which are incomplete and conjoining them poses no serious problems. The difficulty lies in determining whether a constituent following the right conjunct is shared or not. What follows is a brief descrip-

tion of how this might be done.

For a constituent A to be shared by two conjoined constituents C_1 and C_2, there must be rightmost subnodes, N_1 in C_1 and N_2 in C_2, to which A could be grammatically attached. For the parser described in the thesis, this means that there must be packets P_1 and P_2, associated with N_1 and N_2, respectively, each containing a rule whose pattern is satisfied by A and whose action specifies to attach A to the current active node. This suggests that the parser must split into two separate processes: one attempting to attach a trace of A to a rightmost node in C_1 and the other attempting to attach a trace of A to a rightmost node in C_2. If both processes succeed, then A is shared by C_1 and C_2.

While this seems to be a reasonable approach, further investigation of the problem is necessary to determine whether or not it can be implemented without significantly altering the general operation of the parser.

Appendix A. Grammar Rules

The following grammar rules are adequate to parse the sentences used as examples in the thesis. They are not intended to be a complete grammar for English.

Rule to start the parse.

```
{RULE INITIAL-RULE IN NOWHERE
[t] -->
%The parser, by convention, initially calls this rule.%
Create a new root node.
!(setq root c) %so that the structure is around after the
            parse is done.%
Activate cpool,ss-start.}
```

Rules to initiate major clauses.

Packets: SS-START

```
{RULE MAJOR-DECL-S IN SS-START
[=np][=verb] -->
Create a new s node labelled major,decl,fin-s.
Deactivate ss-start.
Activate parse-subj.}

{RULE YES-NO-Q IN SS-START
[=auxverb][=np] -->
Create a new s node labelled quest,ynquest,major,fin-s.
Deactivate ss-start.
Activate parse-subj.}

{RULE IMPERATIVE IN SS-START
Create a new s node labelled imper,major,fin-s.
Deactivate ss-start. Activate parse-subj.}

{RULE HAVE-DIAG PRIORITY:5 IN SS-START
[=*have,tnsless][=np][* is not *and] -->
If 2nd is ns,n3p or 3rd is not verb or 3rd is tnsless
then run imperative next
else run yes-no-q next.}
```

Rules to parse subjects.

Packets: PARSE-SUBJ

{RULE UNMARKED-ORDER IN PARSE-SUBJ
[=np] [=verb] -->
Attach 1st to c as np.
Deactivate parse-subj.
Activate parse-vps.
Activate top-vp.}

{RULE AUX-INVERSION IN PARSE-SUBJ
[=auxverb][=np] -->
%Note that this rule does not reorder the input.%
If 1st is *have then activate have
else if 1st is *be then activate be
else if it is modal then activate modal.
Attach 1st to c as verb.}

Rules to start verb phrases.

Packets: PARSE-VPS, TOP-VP, PARSE-VERB, HAVE, BE, MODAL, TO

{RULE START-VP IN PARSE-VPS
[=verb] -->
Deactivate parse-vps.
Create a new VP node.
Activate parse-verb.}

{RULE VERB IN PARSE-VERB
[=verb] -->
Attach 1st to c as verb.
Deactivate parse-verb.
Activate parse-vps.
If 1st is vp then
  If 1st is *have then activate have
  else if 1st is *be then activate be
  else if 1st is modal then activate modal
  else if 1st is *to then activate to
  else if 1st is mainverb then
    label 1st main;
    label c main;
    Deactivate parse-vps;
    Activate cpool, ss-vp;

```
    if 1st is inf-obj then activate inf-comp
    else if 1st is 2-obj-inf-obj then activate 2-obj-inf-comp
    else if 1st is that-obj then activate that-comp.
  If c is unattached then drop c.}

{RULE ATTACH-TOP-VP IN TOP-VP
[=vp; verb of * agrees with current subject] [**c; * is not vp] -->
Attach 1st to c as vp.
If the vp of c is not complete then set the vp of c to be c
else If c is s then drop c.}

{RULE PERFECTIVE IN HAVE
[* is vp; the verb of * is en] -->
Label c aux, perf.
Attach 1st to c as vp.
If the vp of c is not complete then set the vp of c to be c
else if c is vp then drop c.}

{RULE MAIN-HAVE PRIORITY:15 IN HAVE
[t] -->
%*have must be main verb.%
Label c main.
Deactivate have.
Activate cpool, ss-vp.}

{RULE MODAL IN MODAL
[* is vp; the verb of * is tnsless] -->
Label c aux, modal.
Attach 1st to c as vp.
If the vp of c is not complete then set the vp of c to be c
else if c is vp then drop c.}

{RULE TO IN TO
[* is vp; the verb of * is tnsless] -->
Label c aux, to.
Attach 1st to c as vp.
If the vp of c is not complete then set the vp of c to be c
else if c is vp then drop c.}

{RULE PROGRESSIVE IN BE
[* is vp; the verb of * is ing] -->
Label c aux, prog.
Attach 1st to c as vp.
If the vp of c is not complete then set the vp of c to be c
else if c is vp then drop c.}
```

{RULE PASSIVE IN BE
[* is vp; the verb of * is en] -->
Label c aux, passive.
Attach 1st to c as vp.
If the vp of c is not complete then set the vp of c to be c
else if c is vp then drop c.}

{RULE COPULA PRIORITY:15 IN BE
[t] -->
Label c main.
Deactive be.
Activate cpool, ss-vp.}

Rules to parse VP following main verb.

Packet: SS-VP

{RULE OBJECTS IN SS-VP
[=np] -->
If 1st fits an obj slot of the frame of the of c
then attach 1st to c as np
else run vp-done next.}

{RULE PP-UNDER-VP IN SS-VP
[=pp] -->
If 1st fits a pp slot of the frame of the of c
then attach 1st to c as pp
else run vp-done next.}

{RULE ADVP-UNDER-VP IN SS-VP
[* is adv or advp] -->
If 1st fits an adv slot of the frame of the of c
then attach 1st to c as adv
else run vp-done next.}

{RULE ADJP-UNDER-VP IN SS-VP
[* is adjp or adj] -->
If 1st fits an adjp slot in the frame of the of c
then attach 1st to c as adj
else run vp-done next.}

{RULE VP-DONE PRIORITY:20 IN SS-VP
[**c; * is vp] -->
Drop c.

If c is vp then run vp-done next
else activate ss-final.}

Rules to finalize sentences.

Packets: SS-FINAL, ROOT-FINAL, COMP-S-FINAL

{RULE S-DONE PRIORITY: 20 IN SS-FINAL
[**c; * is s] -->
Drop c.
If c is root then activate root-final.}

{RULE ATTACH-S IN ROOT-FINAL, COMP-S-FINAL
[=s] -->
Attach 1st to c as s.}

{RULE FINAL-PUNC IN ROOT-FINAL
[=finalpunc] [**c; * is root] -->
Attach 1st to c as finalpunc.
Finalize all frames.
Parse is finished.}

Rules to parse noun phrases.

Packets: CPOOL, PARSE-DET, PARSE-ADJECTIVES, PARSE-NOUN,
NP-COMPLETE, NP-DONE

{AS RULE STARTNP IN CPOOL
[=ngstart] -->
Create a new np node.
If 1st is det then activate parse-det
else activate parse-adj, parse-noun.}

{RULE DETERMINER IN PARSE-DET
[=det] -->
Attach 1st to c as det.
Label c det.
Transfer the features indef,def,wh from 1st to c.
Create a new nbar node.
Activate parse-adjectives, parse-noun, np-done.}

{RULE ADJ IN PARSE-ADJECTIVES
[* is adj or adjp] -->

Attach 1st to c as adj.
Deactivate parse-adjectives.}

{RULE NOUN IN PARSE-NOUN
[=noun] -->
Attach 1st to c as noun.
Transfer the features massn,time,ns,npl,n1p,n2p,n3p from 1st to c.
If 1st is any of propnoun,pseudopropnoun, then label c not-modifiable.
Deactivate parse-adjp.
Activate cpool,np-complete. %to parse pp and relative clause modifiers%}

{RULE PP-UNDER-NP IN NP-COMPLETE
[=pp] -->
If 1st fits a pp slot of the frame of c then attach 1st to c as pp.
Deactivate np-complete.
Activate np-done.}

{RULE REL-UNDER-NP IN NP-COMPLETE
[=rel] -->
If 1st fits a rel slot of the frame of c then attach 1st to c as rel.
Deactivate np-complete.
Activate np-done.}

{RULE NP-DONE PRIORITY:15 IN NP-DONE,NP-COMPLETE
[**c; * is np or nbar] -->
Drop c.
If 1st is nbar then activate np-done.}

{RULE ATTACH-NBAR IN NP-DONE
[=nbar] [**c; * is np or nbar] -->
Attach 1st to c as nbar.}

Rules to parse adjective phrases and adverbial phrases.

Packets: CPOOL, PARSE-ADJECTIVES

{RULE ADJP IN CPOOL, PARSE-ADJECTIVES
[=adjmod] [=adj] -->
Create a new adjp node.
Attach 1st to c as adjmod.
Attach 2nd to c as adj.
Drop c.}

{RULE ADVP IN CPOOL

[=advmod] [=adv] -->
Create a new advp node.
Attach 1st to c as advmod.
Attach 2nd to c as adv.
Drop c.}

Rules to parse prepositional phrases.

Packets: CPOOL, PP-NP

{RULE PP IN CPOOL
[=prep] -->
Create a new pp node.
Attach 1st to c as prep.
Activate pp-np.}

{RULE PREP-OBJ IN PP-NP
[=np] -->
Attach 1st to c as np.
Drop c.}

{RULE NO-PREP-OBJ PRIORITY:15 IN PP-NP
[t] -->
Create a new np node labelled trace, not-modifiable.
Drop c. %The np will be bound later%}

Rules to parse that-clauses.

Packets: CPOOL, THAT-COMP

{RULE THAT-S-START PRIORITY:5 IN CPOOL
[=comp,*that] [=np] [=verb] -->
%Marked case%
Label a new s node sec,comp-s,that-s.
Attach 1st to c.
Activate comp-s-final.
Label a new s node fin-s.
Attach 2nd to c as np.
Activate cpool, parse-vps, top-vp.}

{RULE THAT-S-START-1 IN THAT-COMP
[=np] [=verb] -->
Label a new s node sec,comp-s,that-s,fin-s.
Attach 1st to c as np.

Activate cpool, parse-vps, top-vp.}

Rules to parse infinitive clauses.

Packets: CPOOL, INF-COMP

{RULE INF-S-START PRIORITY:5 IN CPOOL
[=*for][=np][=*to] -->
%Marked case%
Label a new s node sec, comp-s, inf-s.
Attach 1st to c as comp.
Label a new s node inf-s.
Attach 2nd to c as np.
Activate cpool, parse-vps, top-vp.}

{RULE INF-S-START-1 IN INF-COMP
[=np] [=*to] [=tnsless] -->
Label a new s node sec, comp-s, inf-s.
Attach 1st to c as np.
Activate cpool, parse-vps, top-vp.}

{RULE CREATE-TRACE-SUBJECT IN INF-COMP, 2-OBJ-INF-COMP
[=*to] [=tnsless] -->
Create an np node labelled trace, not-modifiable.
Drop c.
Activate inf-comp. %In case it is not already active%}

{RULE INSERT-TO IN 2-OBJ-INF-COMP
[=tnsless] -->
%This rule is for verbs like help%
Insert the word 'to' into the buffer before 1st.}

Rule to attach complement sentences to nps.

Packet: CPOOL

{RULE COMP-TO-NP IN CPOOL
[=comp-s] -->
%Bind trace nps.%
If the np of 1st is trace
then if the verb of 1st is ind-obj-binds-trace
%eg. the verbs urge and persuade%
then bind the np of 1st to the np of c

else if the verb of 1st is subj-binds-trace
%eg. the verbs want and promise%
then bind the np of 1st to the np of the current s.
Attach 1st to a new np node labelled comp-np,
not-modifiable as s.
Drop c.}

Rules to parse conjunctions

Packets: all

{RULE CONJOIN-NODES PRIORITY: 25 IN EVERYWHERE
[* is compatible with 3rd][=*and][t] -->
Label a new node (type 1st).
Attach 1st to c as (type 1st).
Attach a new conjp node to c as conjp.
Attach 2nd to c as conj.
Attach 3rd to c as (type 3rd).
Drop c %the conjp node%.
Drop c %the new <node-type> node%.}

{RULE NOTE-RM-NODE PRIORITY: 5 IN EVERYWHERE
[* is not RM-noted][* is *and] -->
Note 1st as an RM node associated with 2nd.
Label 1st RM-noted.}

{RULE PARSE-CONJP PRIORITY: low IN EVERYWHERE
[=*and] -->
Create a new conjp node.
Attach 1st to c as conj.
Activate conj-final, packets in RM set associated with 1st.}

{RULE CHECK-RIGHT-CONJ PRIORITY: high IN CONJ-FINAL
[**c; * has two sons] -->
%The action of this rule will try to find an RM node
whose p/a function can be fulfilled by c's second
son (R).
If there is not a compatible RM node then R is
detached from c and the rule which attached it is
removed from the set of active rules.
If there is a compatible RM node and R can fulfill
the p/a function of the subject of an S node in the
RM-set then R is labelled, detached from c, put back

into the buffer and the priorities of the rules in the
R-packets associated with the S node are temporarily
raised.
If there is a compatible RM node and R cannot fulfill
the p/a function of the subject of an S node in the
RM-set then the son is taken to be the right conjunct
and the RM node, the left conjunct.
C (the conjp node) is dropped into the buffer and then
the left conjunct is detached from its parent, which
is made the new c, and the left conjunct is dropped
into the buffer.%}

{RULE CONNECT-NODE-AND-CONJP PRIORITY: 5 IN EVERYWHERE
[t][=conjp] -->
Label a new node (type 1st).
%type is a function which returns the type of a node%.
Attach 1st to c as (type 1st).
Attach 2nd to c as conjp.
Drop c.}

Key to the Features Used in the Grammar.

adjmod:	adjective modifier (eg. 'very' in 'very careful')
advmod:	adverb modifier (eg. 'very' in 'very carefully')
aux:	vp starting with an auxiliary
auxverb:	auxiliary verb
bad:	marks a constituent which is ungrammatical
comp:	complement markers like 'that' and 'for'
comp-np:	NP node that dominates a complement S
comp-s:	a complement S
copula:	copular verb
decl:	declarative S
def:	definite, of either an article or an NP
det:	determiner
en:	verb with an -en suffix
finalpunc:	final punctuation mark
imper:	imperative S
ind-obj-binds-trace:	verb whose indirect object binds the trace subject of an embedded clause (eg. persuade)
indef:	indefinite, of either an article or an NP
inf:	infinitive verb form
inf-obj:	verb that takes an infinitive complement as its only object (eg. want)

2-obj-inf-obj:	verb taking 2 objects; one is an infinitive complement (eg. persuade)
inf-s:	infinite S
ing:	verb with an -ing suffix
main:	vp starting with the main verb
mainverb:	verb which can be a main verb
major:	major S
massn:	mass noun
modal:	a modal (including will and do)
n1p:	1st person, of nouns
n2p:	2nd person, of nouns
n3p:	3nd person, of nouns and dets
ngstart:	anything that can start an NP (eg. det, noun)
not-modifiable:	np which can take no restrictive modifiers
np:	noun phrase
npl:	plural, of either nouns or quantifiers
ns:	singular, of either nouns or dets
passive:	passive verb
perf:	perfective
prog:	progressive
propnoun:	proper noun
pseudopropnoun:	noun that behaves like a proper noun (eg. 'tommorrow')
quest:	question
rel:	relative clause
sec:	secondary S
subj-binds-trace:	verb whose subject binds the trace subj of an embedded clause (eg. 'want')
that-obj:	verb that takes a tensed complement S (eg. 'believe')
that-s:	tensed complement S
tnsless:	tenseless verb
trace:	np which is a trace
v+13s:	a verb that agrees with a 1st or 3rd person singular noun
v-3s:	a verb that agrees with any noun which is not 3rd person singular noun
v1s:	a verb that agrees only with a 1st person singular noun
v3s:	a verb that agrees with a 3rd person singular noun
vpl+2s:	a verb that agrees with any plural or 2rd person singular noun
vspl:	a verb that agrees with any noun
wh:	a wh- det or an NP with a wh- det
ynquest:	yes/no question

Appendix B. Monitor Programs for the Predicate/Argument Analyser

{ATTACHMENT CRULE VP-VERB VP OVER VERB
Associate a new frame with the lower node.
Associate the frame of lower with upper.
The lower node fills the pred slot of lower.
If the lower node is main and
  the lower node is none of no-subj, passive and
  there is a binding of the np of the s above upper
then it fills the subj slot of lower.}

{ATTACHMENT CRULE VP-VP VP OVER VP
The lower node fills the vp slot of the verb of upper.}

{ATTACHMENT CRULE VP-NP VP OVER NP
The lower node fills an obj slot of the main-verb of upper.}

{ATTACHMENT CRULE VP-PP VP OVER PP
The lower node fills a pp slot of the main-verb of upper.}

{ATTACHMENT CRULE VP-ADVP VP OVER ADVP
The lower node fills an advp slot of the main-verb of upper.}

{ATTACHMENT CRULE VP-ADJP VP OVER ADJP
The lower node fills an adjp slot of the main-verb of upper.}

{ATTACHMENT CRULE NP-DET NP OVER DET
Associate a new frame with upper.
The dummy-predicate for lower fills the pred slot of upper.
The lower node fills the det slot of upper.}

{ATTACHMENT CRULE NP-ADJ/P NP OVER (ADJ,ADJP)
Associate a new frame with upper.
The dummy-predicate for lower fills the pred slot of upper.
The lower node fills an adj slot of upper.}

{ATTACHMENT CRULE NP-N NP OVER N
The lower node fills the subj slot of upper.}

{ATTACHMENT CRULE NP-PP NP OVER PP
The lower node fills a pp slot of upper.}

{ATTACHMENT CRULE NP-REL NP OVER REL
The lower node fills a rel slot of upper.}

{ATTACHMENT CRULE PP-PREP PP OVER PREP
Associate a new frame with lower.
The lower node fills the pred slot of lower.}

{ATTACHMENT CRULE PP-NP PP OVER NP
The lower node fills the obj slot of the prep of upper.}

References

Bresnan, J. [1982] **The Mental Representation of Grammatical Relations**, MIT Press, Cambridge, Mass., 1982.

Fillmore, C. J. [1968] "The Case for Case" in **Universals in Linguistic Theory**, E. Bach and R. T. Harms, eds., Holt, Rinehart, and Winston, N.Y., 1982.

Gazdar, G. [1982] "Phrase Structure Grammar" in **On the Nature of Syntactic Representation**, G. K. Pullem and P. Jacobson, eds., 1982.

Gazdar, G. [1981] "Unbounded Dependencies and Coordinate Structure", *Linguistic Inquiry*, Vol. 12, No. 2., 1981.

Hirst, G. and E. Charniak [1982] "Word Sense and Case Slot Disambiguation", *Proceedings, National Conference on Artificial Intelligence*, Pittsburgh, August 1982.

Hirst, G. [1983] *Semantic Interpretation Against Ambiguity*, Doctoral Dissertation, Department of Computer Science, Brown University, 1983.

Katz, J. J. and J. A. Fodor [1964] "The Structure of a Semantic Theory" in **The Structure of Language**, J. A. Fodor and J. J. Katz, eds., Prentice-Hall, Englewood Cliffs, N.J., 1964.

Marcus, M. [1980] **A Theory of Syntactic Recognition for Natural Language**, MIT Press, Cambridge, Mass., 1980.

Sag, I. [1976] *Deletion and Logical Form*, Doctoral Dissertation, MIT, Cambridge, Mass., 1976.

Schacter, P. [1977] "Constraints on Coordination", *Language*, Vol. 12, No. 2., 1977.

Stockwell, R. P., P. Schacter, and B. Partee [1973] **The Syntactic Structure of English**, Holt, Rinehart, and Winston, N.Y., 1973.