

AN ABDUCTIVE INFERENCE BASED MODEL  
OF HUMAN SENTENCE PARSING

A Thesis

Presented in Partial Fulfillment of the Requirements for  
the Degree Master of Science in the  
Graduate School of The Ohio State University

By

Shravan Vasishth

\* \* \* \* \*

The Ohio State University  
2002

Master's Examination Committee:

Professor John Josephson, Adviser

Professor Richard L. Lewis

Professor B. Chandrasekaran

Approved by

---

Adviser

Department of Computer  
and Information Science

Copyright by  
Shravan Vasishth  
2002

## ABSTRACT

Previous research has found several empirical problems with existing sentence processing theories, e.g., Joshi’s Embedded Pushdown Automaton, Hawkins’ Early Immediate Constituents, Gibson’s Discourse Locality Theory, and Lewis’ Interference and Confusability Theory. This motivates a new model of human sentence parsing based on abductive inference (inference to the best explanation), along with a new complexity metric. The model is a constrained variant of a hybrid top-down and bottom-up parser; it has no lookahead, but has bounded nondeterminism constrained by abductive inferences regarding possible sentence structures. The parsing model and complexity metric account for sentence acceptability data and phrase-by-phrase reading time data from Dutch, German, Hindi, and Japanese sentence processing experiments.

## ACKNOWLEDGMENTS

My greatest debt is to John Josephson and Richard Lewis; they were ideal advisers and guided me skilfully throughout. Rick got me interested in modeling human parsing processes and has been a constant source of inspiration. John's knowledge of abduction was crucial, and several ideas here are due to him. I also thank B. Chandrasekaran for his encouragement, comments, and suggestions.

Many people had helpful suggestions, in particular: Chris Brew, Peter Culicover, Michael Dickey, David Dowty, Anna Feldman, Takao Gunji, Martin Jansche, Keith Johnson, Neal Johnson, Edith Kaan, G.-J. M. Kruijff, Bob Levine, Edson Miyamoto, M. Nakayama, and Gerald Penn. Shari Speer, my Linguistics PhD adviser, saved me from myself many times, and Andrea Vasishth helped considerably with editing.

I thank my outstanding teachers, notably: T. Carlson (in Mathematics), T. Dey, S. Lai, M. Lauria, R. Lewis, S. Parthasarthy, N. Soundarajan, and R. Wenger. Others educated me outside the classroom: Martin Jansche, Daryl McGinnis, and Adam Wolfe. Further afield, Gautam Sengupta (in India) and Michael Wescoat (formerly in Japan), have had a profound impact on my academic life. Perhaps most importantly, thanks go to my renal transplant doctors in India, particularly Dr. S. C. Mehta and Dr. S. C. Tiwari, and to my parents, who gave me a second chance at life.

This research was partly funded by the Center for Cognitive Science, The Ohio State University, and is gratefully acknowledged. Thanks also to Stuart Zweben, the CIS Chair, for finding financial support for me during 2000-2001.

## VITA

March 20, 1964    Born in New Delhi, India

1983              High school certificate  
St. Columba's School, New Delhi

1986-89          BA (Honours), Japanese  
Jawaharlal Nehru University, New Delhi

1986-89          Diplôme de langue française  
Alliance Française, New Delhi

1989-90          Diploma, Japanese Studies Program  
Osaka University of Foreign Studies, Japan

1990-92          Technical translator (Japanese to English)  
Hara Kenzo Patent Law Firm, Osaka, Japan

1992-94          MA, Linguistics  
Jawaharlal Nehru University, New Delhi

1995-96          Research student  
Faculty of Language and Culture, Osaka, Japan

1996-97          Graduate Teaching Assistant  
Faculty of Language and Culture, Osaka, Japan

1997-99, 2001-02    Graduate Research Associate  
Department of Linguistics, The Ohio State University

## PUBLICATIONS

### Research Publications

1. S. Vasishth. “Indefinites and bare singular nominals in Hindi.” *Proceedings of the Chicago Linguistics Society Conference* (2002).
2. S. Vasishth and B. D. Joseph. “Polysemy, Constellations, and *ko* in Hindi.” *Proceedings of the Berkeley Linguistics Society Conference* (2002).
3. S. Vasishth. “Word order, negation, and negative polarity in Hindi.” *Journal of Language and Computation, Volume 3* (2002).
4. S. Vasishth. “An empirical evaluation of sentence processing models: Center embeddings in Hindi.” In *Varia, OSU Working Papers in Linguistics, Volume 56*, edited by Michael Daniels, David Dowty, Anna Feldman, and Vanessa Metcalf. Columbus: The Ohio State University, Department of Linguistics (2001).
5. S. Vasishth and G.-J. M. Kruijff. “Processing as Abduction: A sentence processing model.” In *Proceedings of the Workshop on Linguistic Theory and Grammar Implementation*, edited by Erhard Hinrichs, Detmar Meurers, Shuly Wintner, the 12th European Summer School in Language, Logic, and Information (ESSLLI), Birmingham, UK (2000).

6. S. Vasishth. “Quantificational elements and polarity licensing in Japanese.” *Japanese/Korean Linguistics, Volume 9*, CSLI, Stanford, Cambridge University Press (2000).
7. S. Schwenter and S. Vasishth. “Absolute and relative scalar particles in Spanish and Hindi.” *Proceedings of the Berkeley Linguistics Society Conference* (2000).
8. S. Vasishth. “Monotonicity constraints on negative polarity in Hindi.” In *OSU Working Papers in Linguistics Volume 51*, edited by Mary Bradshaw, David Odden, and Derek Wyckoff. Columbus: The Ohio State University, Department of Linguistics (1998).
9. S. Vasishth. “The NEG-Criterion and negative polarity licensing in Hindi and English.” *Osaka University Journal of Language and Culture, Volume 6* (1997).
10. A. Ohtani and S. Vasishth. “Honorification revisited.” *Osaka University Journal of Language and Culture, Volume 5* (1996).

## FIELDS OF STUDY

Major Field: Computer and Information Science

Area of Specialization: Artificial Intelligence

## TABLE OF CONTENTS

Abstract . . . . .	ii
Acknowledgments . . . . .	iii
Vita . . . . .	iv
List of Figures . . . . .	x
List of Algorithms . . . . .	xii
Chapters:	
1 Introduction . . . . .	1
1.1 What self-center embeddings are, and why they matter . . . . .	1
1.2 Parsing models for self-center embeddings . . . . .	3
2 Four models of human parsing . . . . .	7
2.1 Joshi’s Embedded Pushdown Automaton (1990) . . . . .	7
2.1.1 The Principle of Partial Interpretation and the EPDA . . . . .	9
2.1.2 Summary . . . . .	14
2.2 Hawkins’ Early Immediate Constituents (1994), (1998) . . . . .	14
2.2.1 Some definitions . . . . .	16
2.2.2 The Constituent Recognition Domain and the EIC principle . . . . .	18
2.2.3 Other details regarding parsing decisions . . . . .	20
2.2.4 Summary . . . . .	21
2.3 Gibson’s Dependency Locality Theory (2000) . . . . .	21
2.3.1 Storage cost . . . . .	22

2.3.2	Integration cost . . . . .	23
2.3.3	Other assumptions of the model . . . . .	23
2.3.4	Summary . . . . .	24
2.4	Lewis' Interference and Confusability Theory (2001) . . . . .	25
2.4.1	Interference . . . . .	25
2.4.2	Focus of attention and interference . . . . .	27
2.4.3	Summary . . . . .	27
2.5	Empirical problems with existing parsing models . . . . .	28
2.5.1	Joshi's EPDA . . . . .	28
2.5.2	Hawkins' EIC . . . . .	30
2.5.3	Gibson's DLT . . . . .	32
2.5.4	Lewis' ICT . . . . .	36
3	Human sentence parsing as abduction . . . . .	39
3.1	Introduction . . . . .	39
3.1.1	An informal outline of the model . . . . .	40
3.1.2	Granularity versus breadth of coverage: Some caveats . . . . .	41
3.2	Processing as abduction+deduction: The main proposal . . . . .	42
3.2.1	The underlying competence grammar $\mathcal{G}$ . . . . .	42
3.2.2	The set $\mathcal{H}$ of hypothesis formers . . . . .	43
3.2.3	Some definitions . . . . .	45
3.2.4	The algorithm . . . . .	48
3.2.5	The complexity metric . . . . .	50
3.3	Computational properties of the abductive parser . . . . .	52
4	Empirical coverage of the parsing model . . . . .	55
4.1	Japanese . . . . .	55

4.1.1	Gibson (1998) . . . . .	55
4.1.2	Nakatani et al. (2000) . . . . .	61
4.1.3	Yamashita (1997) . . . . .	63
4.2	Dutch and German . . . . .	65
4.2.1	Dutch: Kaan and Vasić (2000) . . . . .	65
4.2.2	Dutch and German: Bach, Brown, and Marslen-Wilson (1986) . . . . .	67
4.3	Hindi: Vasishth (to appear, 2002) . . . . .	68
5	A Prolog implementation of the model . . . . .	71
5.1	The design of the parser: Some caveats . . . . .	71
5.2	The main call to the program . . . . .	72
5.3	The <code>process/2</code> rule . . . . .	73
5.4	The rule <code>matchverb</code> . . . . .	73
5.4.1	The rule <code>matchverb3</code> . . . . .	77
5.5	The rule <code>abduce_from</code> . . . . .	77
5.6	Some sample runs . . . . .	79
6	Concluding remarks . . . . .	84
6.1	The EPDA and AIM . . . . .	86
6.2	The EIC and AIM . . . . .	86
6.3	The DLT and AIM . . . . .	87
6.4	ICT and AIM . . . . .	88
	Appendix 1: Left-corner parsing and cognitive modeling . . . . .	91
	Appendix 2: Prolog code . . . . .	94
	References . . . . .	106

## LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
2.1 EPDA processing of Dutch dependencies . . . . .	12
2.2 EPDA processing of German dependencies . . . . .	13
2.3 EPDA processing of German mixed dependencies . . . . .	13
2.4 The CRD of the VP in (7a) . . . . .	18
2.5 The CRD of the VP in (7b) . . . . .	18
2.6 Illustration of DLT’s storage cost computation . . . . .	22
2.7 Hindi examples (11a,b) . . . . .	30
2.8 Integration and storage costs for single center embeddings . . . . .	33
2.9 Computation of integration costs at innermost verb . . . . .	33
2.10 Integration and storage costs for IO-dislocated center embeddings (example (17)) . . . . .	34
2.11 Integration costs at innermost verb for IO dislocation (example (17))	34
2.12 Integration and storage costs for DO-dislocated center embeddings (example (18)) . . . . .	35
2.13 Integration costs at innermost verb for DO dislocation (example (18))	35
2.14 Integration and storage costs for center embeddings with or without an adverb . . . . .	35
3.1 Tree representation of schemas . . . . .	43
4.1 Complete derivation history for (25a); total cost = 48 . . . . .	60

4.2	Complete derivation history for (25b); total cost = 39 . . . . .	61
4.3	Complete derivation for (26a); total cost = 38 . . . . .	62
4.4	Complete derivation for (26b); total cost = 25 . . . . .	62
4.5	Derivation history for (27a); total cost = 38 . . . . .	63
4.6	Complete derivation history for (27b); total cost = 48 . . . . .	63
4.7	Derivation for (28) . . . . .	65
4.8	Derivation for (29a); total cost = 50 . . . . .	66
4.9	Derivation for (29b); total cost = 28 . . . . .	66
4.10	Derivation for Dutch crossed embedding (30a); total cost = 41 . . . . .	68
4.11	Derivation for German nested embedding (30b); total cost = 41 . . . . .	68
4.12	Derivation for Hindi examples (31a,b) . . . . .	69
5.1	The main call to the parser . . . . .	72
5.2	The call to <code>process/2</code> . . . . .	74
5.3	Processing in the case where a matrix verb is seen . . . . .	75
5.4	Processing a non-matrix verb . . . . .	76
5.5	The <code>matchverb</code> rule: The simple case . . . . .	76
5.6	The rule <code>matchverb3</code> . . . . .	77
5.7	The rule <code>abduce_ from</code> . . . . .	78
5.8	The rule <code>abduced_ hyps</code> . . . . .	78
5.9	The rule <code>get_ args</code> . . . . .	79
5.10	The rules for incremental search . . . . .	80
5.11	Trace showing working memory changes for (36) . . . . .	81
5.12	Trace showing working memory changes for (37) . . . . .	82
5.13	Trace showing working memory changes for (38) . . . . .	83
6.1	Schematic illustration of interference . . . . .	88

A1.1 The three main types of parsing strategy . . . . .	92
A1.2 Space complexity for the constructions in (41) . . . . .	93

## LIST OF ALGORITHMS

<u>Algorithm</u>	<u>Page</u>
1 Main loop of the parser . . . . .	49
2 The main function <b>Process</b> ( $S$ ) . . . . .	49

## CHAPTER 1

### INTRODUCTION

#### 1.1 What self-center embeddings are, and why they matter

Self-center-embedding constructions (hereafter, SCEs) are grammatical structures in which a constituent occurs medially within a larger instance of the same kind of syntactic category. Examples from English are sentences like the single embedding (1a), the double embedding (1b), and the triple embedding (1c); see (Sampson, 1996, 17,21,22).

- (1) a. Don't you find [that sentences [that people you know produce] are easier to understand]?
- b. The odds [that your theory will be in fact right, and the general thing [that everybody's working on] will be wrong,] are low.
- c. Your report today [that any Tory constituency party [failing [to deselect its MP], should he not vote in disobedience with a prime ministerial diktat,] might itself be disbanded], shows with certainty that ...an 'elective dictatorship' is now with us.

It is well-known since Chomsky and Miller (1963) that the greater the amount of embedding in an SCE, the more difficult it is for English speakers to process.

Chomsky also observed that unbounded SCEs place human languages outside the class of languages recognizable by a finite state automaton (or, equivalently, regular grammars; see (Hopcroft & Ullman, 1979)), and that natural language must at least belong to the class of context-free languages. Of course, this conclusion presupposes that sentences like (1) are in fact grammatical English sentences. Chomsky argued that they cannot be ruled out by English grammar if recursion in general is allowed; rather, they are grammatical but hard (or impossible) to comprehend. Call this the **GRAMMATICALITY ASSUMPTION**.

Chomsky's argument for making this assumption was that language has the property of recursion, and therefore there is no *principled* way of preventing a structure where clauses are successively embedded inside other clauses (one can, of course, arbitrarily put an upper bound on the number of embeddings allowed by the grammar; see (Reich, 1969) for one such attempt). However, there is more than one kind of recursion, and only recursive structures that result in self-embeddings are problematic for finite state automata and place the grammar outside their expressive power. For example, Kleene-star recursion appears in language in the form of right-branching constructions. These are structures like *The student said that the professor told him that . . .*, which can be continued indefinitely with more and more *that* clauses appended to the right. Kleene-star recursion in language is, of course, expressible by finite state automata, since no record needs to be kept of how many right branchings occurred (Miller & Chomsky, 1963, 470). Chomsky's point was that natural language grammar should be able to express recursion *in general*, and since finite state automata cannot fully characterize all the recursive properties of language, we must look to a more powerful kind of grammar (e.g., context-free grammar).

Reich (1969) and others have argued that the grammaticality assumption is not valid (see (Lewis, 1996b) for an overview), and that SCEs are not hard to process, but simply ungrammatical if the level of embedding goes beyond one. One problem with this view is that double and triple SCEs occur quite freely in naturally occurring text corpora (Roeck et al., 1982), (Sampson, 1996), (Korthals, 2001). Another problem is that, even if we were to ignore such corpus evidence, the question appears to be undecidable, since in general there appears to be no way to distinguish between the grammaticalization of a processing constraint and the processing constraint itself. That is, processing constraints could simply become hard-wired into the competence grammar over time; see (Hawkins, 1994) for a detailed defence of such a grammaticalization view. In fact, Reich himself states that his model “...is not only a linguistic model, but also a psychological model [of] short-term memory in the *production* and *comprehension* of sentences” (Reich, 1969, 841) (my emphasis). If Reich’s is also a model of human short-term memory, how do we distinguish the linguistic constraint from the constraint on short-term memory?

Given that single, double, and even triple SCEs do occur in natural language corpora, in this thesis I will assume that SCEs are grammatical but difficult to process compared to other syntactic structures like right-branching constructions.

## 1.2 Parsing models for self-center embeddings

A great deal of research on models of human parsing has focused on processing difficulty associated with ambiguity and garden-path sentences, inspired by, e.g., (Bever, 1970), (Frazier, 1979), although one sees an occasional nod to the center embedding problem (notable exceptions are Gibson’s and Lewis’ work, which provide

detailed accounts of both ambiguous and unambiguous sentences). Ambiguity and garden-pathing are certainly important questions for psycholinguistic research, but they are not the only ones.

In this thesis I focus exclusively on the modeling of moment-by-moment processing difficulty associated with center embeddings in Dutch, German, Japanese, and Hindi. I chose center embeddings and this set of languages for two reasons: (a) we have a coherent body of experimental results that allow us to evaluate models of sentence processing (for Dutch, see (Dickey & Vonk, 1997), (Kaan & Vasić, 2000); for Dutch and German, see (Bach, Brown, & Marslen-Wilson, 1986); for Japanese, see (Lewis & Nakayama, 2001), (Babyonyshev & Gibson, 1999), (Uehara & Bradley, 1996); and for Hindi, (Vasishth, to appear, 2002)); and (b) even for such a small set of languages the processing facts cannot be adequately explained by the existing models. Examples from each type are shown below (the examples are taken from the above-mentioned studies). Note that the German example is fairly unnatural for native speakers, but this example is only for expository purposes; natural counterparts can be, and have been, found in corpora, as mentioned earlier.

- (2) a. (dat) Aad Jantje de lerares de knikkers liet helpen opruimen  
 that Aad Jantje the teacher the marbles let help collect  
 ‘(that) Aad let Jantje help the teacher collect the marbles.’
- b. (dass) die Männer haben Hans die Pferde füttern lehren  
 that the men have Hans the horses feed teach  
 ‘(that) the men have taught Hans to feed the horses.’
- c. Keiko-ga Tadashi-ga Kenji-o kiraida-to omotteiru  
 Keiko-nom Tadashi-nom Kenji-acc hates-comp thinks  
 ‘Keiko thinks that Tadashi hates Kenji.’
- d. Siitaa-ne Hari-ko kitaab khariid-neko kahaa  
 Sita-erg Hari-dat book buy-inf-acc said  
 ‘Sita told Hari to buy a/the book.’

One question that has received much attention is: why are such sentences difficult for humans to process? Pre-theoretically, it is clear that the noun phrases (NPs) must somehow be temporarily stored/encoded in working memory<sup>1</sup> until one or more verbs necessitate the NPs' retrieval (from working memory) and subsequent integration with the verb(s). This observation, along with the fact that working memory is resource-bounded (Miller, 1956), suggests that inherent constraints on working memory are the source of processing difficulty (Yngve, 1960). (Miller & Chomsky, 1963), (Gibson, 1998), (Gibson, 2000), (Hawkins, 1994), (Hawkins, 1998), (Hudson, 1996), (Lewis & Nakayama, 2001).

Four theories of sentence processing are particularly interesting in this regard: Joshi's Embedded Pushdown Automaton (EPDA) (Joshi, 1990), Gibson's Dependency Locality Theory (DLT) (Gibson, 2000), Lewis' Interference and Confusability Theory (ICT) (Lewis & Nakayama, 2001), and Hawkins' Principle of Early Immediate Constituents (EIC) (Hawkins, 1994), (Hawkins, 1998). These theories appeal to working memory constraints for explaining the processing of syntactic structures like SCEs, and are "universalist" in that they aim for cross-linguistic applicability. The above theories are attractive for at least four reasons: (a) these are information-processing theories of parsing, that is, they all present specific views of the human mind as a system that stores, manipulates, and processes information (this is interesting because they make falsifiable hypotheses about how the mind functions when it engages in parsing); (b) they have either been implemented as computer programs (Lewis), or are specified precisely enough to allow such an implementation (Joshi, Hawkins, Gibson); (c) except perhaps for Joshi's EPDA, they

---

<sup>1</sup>Working memory or short-term memory, is "... a short-duration system which small amounts of information are simultaneously stored and manipulated in the service of accomplishing a task" (Caplan & Waters, 1999).

are able to account for many processing facts in a wide array of languages; and (d) all of these theories aim for fine granularity, that is, they aim to provide an account for phrase-by-phrase processing difficulty as a sentence is processed.

Fine granularity is important if the goal is to build strongly equivalent models of human linguistic perception, as opposed to robust, wide-coverage parsers for natural language processing in real-world applications. There is a tension between these two goals, and at least at present, the one precludes the other. This is because we currently do not understand human parsing mechanisms in sufficient detail to allow their application in robust parsing; the tendency in real-world applications currently is to adopt efficient computational techniques which may not have any correlate to psychological processes (e.g., memoization, which is used to avoid repeating already-completed computations). Perhaps, when enough is known about human parsing processes, it will be possible to have both strong equivalence and robust parsing capability. I return to this issue in Section 3.1.2, but for the moment, suffice to say that this thesis is concerned with the strong equivalence problem.

The rest of the thesis is devoted to this goal, and is structured as follows. Chapter 2 describes the four models mentioned above and discusses some empirical problems they face. These problems motivate the new model of human parsing presented in Chapter 3. Then the empirical coverage of the proposed model is discussed in Chapter 4. Chapter 5 presents a Prolog implementation of the model. Chapter 6 concludes the thesis with a discussion about the similarities and differences between the proposed model and the existing ones, and ways in which the model might be extended.

## CHAPTER 2

### FOUR MODELS OF HUMAN PARSING

The difficult part ... is to figure out how many of the details are needed to give realistic predictions. ... models tend to be so general that they cannot make predictions about particular systems, or so detailed that they merely rephrase what is already known about a system.

*Ants at Work*, by Deborah Gordon, p. 149

In this chapter, I present the details of each of the four models mentioned earlier, and then summarize the empirical problems associated with each, particularly in connection with the facts about center embeddings.

#### 2.1 Joshi's Embedded Pushdown Automaton (1990)

Joshi (1990)<sup>1</sup> presents a computational model of processing based on the results of Bach et al. (1986), who showed that Dutch self-center embeddings (SCEs) were easier to process for native Dutch speakers than German SCEs are for native German speakers. Briefly, subjects were asked to give an acceptability rating to each sentence, and were also tested on comprehension. The sentences were matched across German and Dutch, and varied from simple, easy-to-understand sentences, to SCEs with three levels of embedding. The key finding was that, compared to the German subjects

---

<sup>1</sup>I discuss the 1990 version even though (Rambow & Joshi, 1994) is the more recent one; this is because the 1990 version constitutes a proper subset of the later one, and is simpler to present.

evaluating German embeddings, Dutch subjects rated the Dutch embeddings to be better, and performed better on the comprehension test.<sup>2</sup> Examples of Dutch and German SCEs are shown below:

- (3) a. Jan Piet Marie zag laten zwemmen  
Jan Piet Marie saw make swim  
'Jan saw Piet make Marie swim.'
- NP1 NP2 NP3 V1 V2 V3
- b. ...dass Hans Peter Marie schwimmen lassen sah  
...that Hans Peter Marie swim make saw  
'...that Hans saw Peter make Marie swim.'
- NP1 NP2 NP3 V3 V2 V1

The Dutch SCEs are called “crossed dependencies” because of the fact that the dependencies between the verbs and the subjects form crossing chains, and the German SCEs are called “nested dependencies” since each embedded dependency is nested within another dependency.

Bach et al. (1986) follow (Evers, 1975, 58-59) in arguing that the pushdown automaton (PDA) cannot be the universal basis for the human parsing mechanism if Dutch crossed dependencies are easier to process than German serial dependencies. The problem for the PDA is that in the case of Dutch, the first verb encountered has to be integrated with the first NP, but this NP is at the bottom of the stack, and therefore not immediately accessible. The argument thus is that if Dutch is easier to process than German, it should be easier (not impossible, as in the case of the PDA) to access the first NP once the first verb is encountered.

---

<sup>2</sup>A common objection to the Bach et al. paper is that one cannot really compare two different languages directly. The main point of the Bach et al. paper was that one can in fact do this. There is clearly a possible confound in such a study, but to the extent that this cross-linguistic approach is valid, the Dutch-German contrast is relevant to sentence processing questions. I will not attempt to resolve the question of its relevance *per se* in this thesis.

### 2.1.1 The Principle of Partial Interpretation and the EPDA

Joshi proposes a PRINCIPLE OF PARTIAL INTERPRETATION, which is a formalization of an observation in (Bach et al., 1986) regarding the processing of German nested dependencies. The problem with German nested dependencies is that the final NP can combine with the first verb encountered, but there is no structure for it to be attached to, since none has been built so far. Joshi formalizes this as a constraint on his processing model (Joshi, 1990, 4-5):

1. The structure should be a properly integrated structure with respect to the predicate-argument structure (i.e., only predicates and arguments that go together should be integrated: *ad hoc* packaging of predicates and arguments is disallowed), and there should be a place for it to go if it is expected to fit into another structure (i.e., the structure into which it will fit must have been popped already).
2. If a structure which has a slot for receiving another structure has been popped, then the structure that will fill this slot will be popped next.

Joshi then develops an embedded PDA (EPDA) and shows that it can handle the Dutch and German processing facts. The significance of this is that EPDAs are equivalent to the syntactic formalisms tree-adjoining grammar (Abeillé & Rambow, 2000), head-driven phrase structure grammar (Pollard & Sag, 1994), and combinatorial categorial grammar (Steedman, 2000), all of which are capable of providing syntactic analyses for crossed and nested dependencies. By the term ‘equivalence’, Joshi means the following: for any tree-adjoining grammar (TAG)  $G$ , there is an EPDA  $M$  such that the language  $L(M)$  recognized by  $M$  is exactly the

language  $L(G)$  generated by  $G$  (that is,  $L(G) = L(M)$ ), and (conversely) for any EPDA  $M'$ , there is a TAG  $G'$  such that  $L(M') = L(G')$ .

As for the connection between linguistic grammars and the corresponding EPDA, Joshi puts it as follows: “If we take the processing account as our primary concern the automaton models (such as EPDAs) satisfying some natural constraints are the natural objects to investigate. This raises a question about the status of the grammars associated with the automaton models (EPDAs) in our case, and more interestingly the ‘linguistic’ grammars themselves!” The implication seems to be that the grammars associated with the EPDA may themselves be the linguistic grammars. In other words, performance constrains competence (this view is presented in a slightly different form in more recent work by Hawkins (1998)– see Section 2.2). As Lewis (personal communication) points out, there is an additional, crucial component to the EPDA, the complexity metric (discussed later in this section); this does not derive directly from the EPDA but derives rather from (implicit) claims about space constraints on working memory. One must therefore conclude that Joshi’s claim is really that the EPDA, along with its complexity metric, is an adequate model of human sentence parsing, and that the EPDA is associated with a grammar that can provide a linguistic analysis of the given syntactic structure.

In the following discussion of Joshi’s model, I assume a working knowledge of PDAs (Hopcroft & Ullman, 1979, 107-124). In an EPDA, the pushdown store is a sequence of stacks, and new stacks may be created above or below (to the left or right) of the current stack. The specific behavior of EPDAs described below is based on (Joshi, 1990).

1. **Stack head:** This is always at the top symbol of the top stack. If the stack head ever reaches the bottom of a stack, then the stack head automatically

points to the top of the stack below (or to the left) of the current stack, if there is one.

2. **Transition function  $\delta'$ :** Given an input symbol, the state of the finite control and the stack symbol, this specifies (a) the new state; (b) whether the current stack is pushed or popped; and (c) new stacks to be created above or below the current stack.

$$\delta'(\text{input symbol, current state, stack symbol}) =$$

$$(\text{new state, } sb_1, sb_2, \dots, sb_m, \text{ push/pop on current stack, } st_1, st_2, \dots, st_n)$$

where  $sb_1, sb_2, \dots, sb_m$  are the stacks introduced below the current stack, and  $st_1, st_2, \dots, st_n$  are the stacks introduced above it.

Note that during each move, push/pop is carried out on the current stack, and pushes on the newly created stack(s).

Next, I illustrate processing of the Dutch crossed dependency sequence (NP1 NP2 NP3 V1 V2 V3) in Figure 2.1. The column “Stack sequence” contains the newly created stacks, “Stack” is the initial stack, and the column “Pop action” shows how the interpretation is incrementally built up. Finally, “No. of (input) items” lists a number that Joshi uses as a complexity measure to account for the difference in processing Dutch and German— this just involves adding up the total number of input items in the EPDA at each move, and looking at the largest number (in the Dutch case, 3).

The parse proceeds as follows. First, NP1 is read in and pushed on to the current stack, the same goes for NP2 and NP3. Then NP3, NP2, and NP1 are successively popped out of the current stack and pushed into sequences of stacks at the left of the current stack. After that, each NP is popped out of the EPDA and

Input head at	Stack sequence	Stack	Pop action	No. of items
NP1				0
NP2		NP1		1
NP3		NP1 NP2		2
V1		NP1 NP2 NP3		3
V1		NP3		3
V1		NP1 NP2		3
V1	NP3	NP1		3
V1	NP3	NP2		3
V2	NP3	NP2	V1(NP1,S1)	2
V3	NP3	NP3	V2(NP2,S2)=S1	1
			V3(NP3)=S2	0

Figure 2.1: EPDA processing of Dutch dependencies

incrementally builds up the predicate-argument structures starting with V1 up to V3. The complexity, measured by the number of items in the stack, never goes beyond 3.

The German case, where the order of NP and V sequences is NP1 NP2 NP3 V3 V2 V1, is handled as shown in Figure 2.2. In each case,  $V^*n$  is a possibly underspecified structure encoding  $Vn$  and its argument(s) ( $NPn$  and possibly also S). That is,  $V^*3 = V1(NP3)$ ,  $V^*2 = V2(NP2, S2)$ , and  $V^*1 = V3(N1, S1)$ . Note that the maximum number of input items in this case is 6 (since there are six items in the stack: V3, NP3, V2, NP2, V1, and NP1), which is higher than in Dutch crossed dependencies.

Joshi also discusses the case of mixed dependencies in German, where the sequences in the order NP1 NP2 NP3 V1 V3 V2. This kind of dependency is claimed to have a complexity intermediate between that for crossed and nested dependencies (presumably due to the larger number of total steps involved in mixed dependencies). In such a case, the EPDA behaves exactly like that for nested dependencies in German until we reach V1. Then it must behave like the EPDA for crossed dependencies. A schematic view is shown in Figure 2.3:

Input head at	Stack sequence	Stack	Pop action	No. of items
NP1				0
NP2		NP1		1
NP3		NP1 NP2		2
V3		NP1 NP2 NP3		3
V3		NP1 NP2		4
V3	V*3	NP1		5
V3	V*3 V*2			6
V3	V*3 V*2 V*1			6
V2	V*3 V*2		V1(NP1,S1)	4
V1	V*3		V2(NP2,S2)=S1	1
			V3(NP3)=S2	0

Figure 2.2: EPDA processing of German dependencies

Input head at	Stack sequence	Stack	Pop action	No. of items
NP1				0
NP2		NP1		1
NP3		NP1 NP2		2
V1		NP1 NP2 NP3		3
V1		NP1 NP2		3
V1	NP3	NP1		3
V1	NP3 NP2			3
V1	NP3 NP2			3
V3	NP3 NP2		V1(NP1,S1)	2
V2	NP3 V3			3
	NP3 V3		V2(NP2)=S1	2
	NP3		V3(NP)	1
			NP3	0

Figure 2.3: EPDA processing of German mixed dependencies

Note that when V3 is popped out, its argument (NP) is uninstantiated (see the penultimate line in Figure 2.3). This is only instantiated when NP3 is popped out in the final move. Another important point is that, when the input head is at V2, the preceding V3 has been inserted to the *left* of NP2 by creating a new stack behind the stack holding NP2, and inserting V3 into this new stack. These moves are driven by the principle of partial interpretation: the V3 cannot attach to any structure and therefore must be stored.

### 2.1.2 Summary

In sum, Joshi's main claim is that a variant of the pushdown automaton, along with a complexity metric, can model performance, and since a competence grammar is associated with such an automaton, there is a direct connection between performance and competence models. Although Joshi does not make this explicit, the implicit claim seems to be that performance constraints may determine competence grammars.<sup>3</sup>

## 2.2 Hawkins' Early Immediate Constituents (1994), (1998)

Although Hawkins (1994; 1998) presents EIC as a theory of human sentence processing, it differs from the other theories discussed here in that its central claim is that constraints on processing (such as constraints on extraposition, scrambling,

---

<sup>3</sup>In a more recent version of the EPDA account (Rambow & Joshi, 1994), Rambow and Joshi suggest that a linguistic (or competence) grammar cast in TAG can be used to derive an appropriate bottom-up embedded pushdown automaton (BEPDA). This approach reduces the problem of building a performance model to the question of building a competence grammar. This detail turns out to be orthogonal to the main problems with the EPDA and the BEPDA, which I discuss later in this chapter. I only mention this issue here for completeness.

etc.) determine the grammars of different languages. Competence grammar is an “evolutionary movement” towards an ideal specified by the theory. Under this view, the grammar only contains general principles of linguistic structure (such as the structure of verb phrases); there are no innate and parameterized universals for word order in the competence grammar (cf. (Chomsky, 1965)); rather, performance constraints become “conventionalized” (grammaticalized) to determine preferences for optimal and sub-optimal word orders.

An appealing feature of this processing-driven view of grammar is that it provides true explanations of grammatical phenomena, as opposed to merely recoding some observed fact as an innately present principle of language. For example, it is known (Hawkins, 1994, 257) that when verbs (Vs) modify prepositional phrases (PPs), cross-linguistically the preferred order is one where the V and head of the PP are adjacent (that is, the orders [V [P NP]] and [[NP P] V] are preferred over [V [NP P]] and [[P NP] V]). A typical syntactic analysis would be to reassert as a principle something that we already know: languages are parameterized to be head-initial or head-final, and that any deviances from the head-parameter are “marked” and therefore dispreferred. The EIC provides a different account whereby the preferred orders are a result of immediate-constituent-to-word ratios (these are discussed below in detail).<sup>4</sup>

To give another example, Chomsky and others have argued that center embeddings like (4) are grammatical but hard to process. Under Hawkins’ view, English center embeddings like (4) are not, as Chomsky would claim, grammatical but hard to process, but hard to process and therefore ungrammatical.

---

<sup>4</sup>Given a syntactic tree for any natural language sentence, a CONSTITUENT is any part of the sentence that occurs under a node in the tree, and a node *A* is an IMMEDIATE CONSTITUENT of a node *B* if *B* immediately dominates *A*. Any node *B* IMMEDIATELY DOMINATES another node *A* iff no other node intervenes between *A* and *B*.

- (4) The mouse that the rat that the cat chased bit died.

This center-embedding construction is assumed to be ruled out by the English grammar through a grammaticalization of the processing difficulty associated with these structures. By contrast, the grammars of other languages, like German and Japanese, do allow center embeddings, but this is because the processing difficulty associated with center embeddings is not as great in these languages as in English (Hawkins, 1994, 7-12).

In the following sections, I present the theory with the help of some motivating examples. The discussion below is based on (Hawkins, 1998).

### 2.2.1 Some definitions

Hawkins' theory relies on the notions of STRUCTURAL DOMAINS, STRUCTURALLY RELATED NODES, and STRUCTURAL INTEGRATION in order to specify the source(s) of parsing difficulty. I begin by defining these, and then motivate the relevance of these notions with examples.

STRUCTURAL DOMAINS (SDs):

Grammatically or psycholinguistically significant subsets of STRUCTURALLY RELATED NODES of trees dominated by a given constituent.

STRUCTURALLY RELATED NODES:

Two nodes are structurally related if they are sisters, if one dominates the other, or if one is a sister of some third node that dominates the other.

STRUCTURAL INTEGRATION:

A node A structurally integrates another node B iff A c-commands B.<sup>5</sup>

For example, consider (5):

(5) ... [<sub>NP</sub> the woman<sub>i</sub> [<sub>S'</sub> whom<sub>i</sub> John saw O<sub>i</sub>]]...

The SD of the relative clause *whom John saw* consists of all the nodes dominated by the NP node that structurally integrate the empty position O<sub>i</sub>.

The more nodes in an SD, the greater its complexity. For example, consider (6). Here, the empty position O<sub>i</sub> is more deeply embedded and has more nodes c-commanding it in the SD than in the SD shown in (5); (6) is therefore predicted to be harder to process than (5).

(6) ... [<sub>NP</sub> the woman<sub>i</sub> [<sub>S'</sub> whose<sub>i</sub> work John studies [<sub>NP</sub>O<sub>i</sub>...]]]...

Hawkins quantifies the relative complexity of SDs by defining a complexity metric which relies on (i) the notion of CONSTITUENT RECOGNITION DOMAIN (CRD); (ii) the principle of EARLY IMMEDIATE CONSTITUENTS (EIC); (iii) the MOTHER NODE CONSTRUCTION principle; and (iv) the parsing axiom of CONSTRUCTIBILITY. In the next section, I first informally motivate each of the concepts, and then provide the precise definitions.

---

<sup>5</sup>C-command is defined over trees: A node A c-commands another node B if neither DOMINATES the other but every node that dominates A also dominates B. A node X dominates another node Y if there is a uniformly downward path from X to Y.

## 2.2.2 The Constituent Recognition Domain and the EIC principle

Informally, the CRD of a given phrase (such as a VP) is the set of words that must be scanned before that phrase and its immediate constituents can be recognized. To illustrate, consider (7a,b) and the corresponding Figures 2.4 and 2.5.

- (7) a. He [<sub>VP</sub> [<sub>V</sub> donated] [<sub>PP</sub> to charity] [<sub>NP</sub> the beautiful desk dating from the early Victorian period]].
- b. He [<sub>VP</sub> [<sub>V</sub> donated] [<sub>NP</sub> the beautiful desk dating from the early Victorian period] [<sub>PP</sub> to charity]].

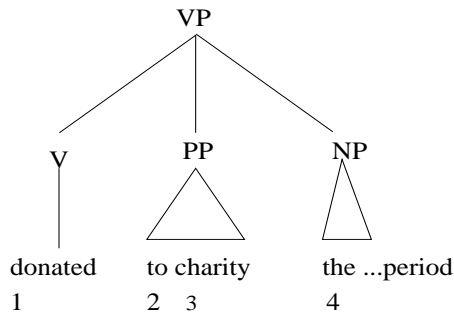


Figure 2.4: The CRD of the VP in (7a)

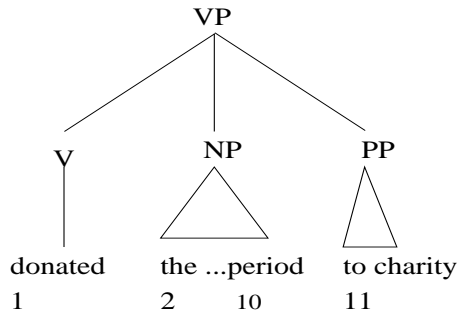


Figure 2.5: The CRD of the VP in (7b)

In (7a), the VP's CRD consists of the four words that need to be scanned before the VP is recognized, and in (7b) the VP's CRD consists of eleven words; see Figures 2.4 and 2.5.

The CRD, a set of words, is the basis for making parsing decisions, and parsing difficulty is quantified by the ratio of immediate constituents (ICs) to non-ICs in the CRD. A less accurate but simpler version of this is to simply take the IC-to-word ratio (Hawkins, 1994, 74-77). For example, in the CRD of the VP in (7a) there are three ICs (V, PP, NP) and 4 words, so the IC-to-word ratio is  $3/4$ . The VP's CRD in (7b) also has 3 ICs, but it now has 11 words, so its IC-to-word ratio is  $3/11$ . The overall complexity of a sentence is taken to be the mean of the complexities of each phrase in the sentence. These ratios quantify the relative differences in parsing difficulty: the higher the ratio, the easier the processing. That is, the fewer the words per CRD, the easier the processing. This preference for fewer words per CRD relates to constraints in human working memory, which presumably prefers buffering fewer words rather than more in order to recognize a CRD (Hawkins, 1998, 732).

Formally, the CRD is defined as follows:

CONSTITUENT RECOGNITION DOMAIN (CRD):

The CRD for a phrasal mother node M consists of the set of terminal and non-terminal nodes that must be parsed in order to recognize M and all ICs of M, proceeding from the terminal node in the parse string that constructs the first IC on the left, to the terminal node that constructs the last IC on the right, and including all intervening terminal nodes and the non-terminal nodes that they construct.

The claim that higher ratios lead to easier processing is codified as the principle of Early Immediate Constituents (EIC).

### EARLY IMMEDIATE CONSTITUENTS (EIC):

The human parser prefers linear orders that maximize the IC-to-nonIC (or IC-to-word) ratios of Constituent Recognition Domains. Orders with the most optimal ratios will be preferred over their non-optimal counterparts in the unmarked case; orders with non-optimal ratios will be more or equally preferred in direct proportion to the magnitude of their ratios. For finer distinctions, ratios can be measured left-to-right.

Parsing decisions are based solely on the principle of EIC; pragmatic factors such as given-ness or new-ness play no role per se; in this sense it is a syntax-only theory (Hawkins, 1998, 751-759).

### 2.2.3 Other details regarding parsing decisions

Apart from recognizing a phrase M based on the CRD of M, the parser also has to build the corresponding node for M in the tree. The next principle, Mother Node Construction, stipulates that this node is built as soon as a word appears that uniquely determines the existence of M. For example, a verb uniquely determines a VP; therefore, seeing a verb is necessary and sufficient information for the parser to build a VP.

### MOTHER NODE CONSTRUCTION (MNC):

In the left-to-right parsing of a sentence, if any word<sup>6</sup> of syntactic category C uniquely determines a phrasal mother node M, in accordance with the PS rules of the grammar, then M is immediately constructed over C.

---

<sup>6</sup>This includes bound morphemes, such as suffixes marking infinitival verbs (Hawkins, 1994, 359-379), (Hawkins, 1998, 733).

The parse succeeds if a phrasal node M has been built in accordance with MNC. Hawkins states this as an axiom:

AXIOM OF CONSTRUCTIBILITY:

For each phrasal node M there will be at least one word of category C dominated by M that can construct M on each occasion of use.

The above discussion means that the model essentially uses a head-driven parser: a head of a phrase must be encountered before the corresponding phrasal node is built. For example, a verb must be seen before subtree rooted at a verb phrase node is constructed.

#### 2.2.4 Summary

To summarize, Hawkins' view is that the processing principle of Early Immediate Constituents can account for both real-time processing facts and word order preferences cross-linguistically; the grammar of a language is a result of the grammaticalization of performance constraints. The most compelling aspect of Hawkins' theory is that it can account for diverse word-order facts from an impressive array of languages (Hawkins, 1998, 734-741): English, Hungarian, Romanian, Polish, German, Greek, Japanese, and Korean, to name a few.

### 2.3 Gibson's Dependency Locality Theory (2000)

The Dependency Locality Theory (DLT) aims to account for processing difficulty in both ambiguous (garden-path) structures and unambiguous ones, such as center

embeddings. In this discussion, I will focus on the DLT account for center embeddings. The discussion in this section is based on (Gibson, 2000).

The DLT assumes that during the course of sentence parsing, computational resources in working memory are needed for two aspects, STORAGE of the structure built up thus far; and INTEGRATION of the current word into the structure built up thus far. The DLT predicts processing complexity based on these two components and a cost metric. Cost is measured in MEMORY UNITS (MUs). There is a STORAGE COST and an INTEGRATION COST. I describe these two components next.

### 2.3.1 Storage cost

Storage cost is computed as follows: 1 MU is associated with each syntactic head required to complete the current input as a grammatical sentence. For example, as shown in Figure 2.6, for the first word, *The*, a noun and a verb are needed to complete the sentence grammatically, so the storage cost is 2 MUs. At the second word, *reporter*, only a verb is needed, so the storage cost is 1 MU. For the next word, *disliked*, one NP is needed, so the cost is 1 MU. The cost for the next word, *the*, is also 1 MU, since only a noun is needed to complete the input string as a grammatical sentence. The last word incurs no cost since it can complete the sentence.

Input words:	The	reporter	disliked	the	editor
Syntactic heads needed:	Noun, Verb	Verb	NP	Noun	-
Storage cost (MUs):	2	1	1	1	0

Figure 2.6: Illustration of DLT’s storage cost computation

### 2.3.2 Integration cost

The cost of integrating a current word  $A$  with an existing structure  $S$  depends on the LOCALITY, i.e., the distance, between them. More precisely, the cost depends on the complexity of the computations that took place between  $A$  and  $S$ . These computations are assumed to be linearly related to the number of discourse referents (DRs) introduced between the two items.<sup>7</sup> Thus, the cost of integrating  $A$  with  $S$  is quantified in terms of the number of intervening DRs. The unit of measure for integration cost is an Energy Unit (EU).

### 2.3.3 Other assumptions of the model

Apart from storage cost and integration cost, the other factors that are assumed to affect comprehension difficulty are:

1. The frequency of the lexical item being integrated: The lower the frequency, the greater the processing difficulty.
2. The contextual plausibility of the resulting structure: The less plausible the final structure, the greater the processing difficulty.
3. The discourse complexity of the final structure: Nonfocused entities and elements introduced using definite descriptions are less accessible (and are therefore harder to process) than focused entities. Least accessible of all are

---

<sup>7</sup>Discourse referents are defined as follows: “A discourse referent is an entity that has a spatiotemporal location so that it can later be referred to with an anaphoric expression, such as a pronoun for NPs, or tense on a verb for events” (Gibson, 2000, 103). It is important to note that Gibson does not regard discourse referents as the only measure of distance: “Processing other kinds of elements [other than new discourse referents] also probably causes measurable increments in structural integration cost” (Gibson, 2000, 107). However, the DLT has nothing more to say about other measures of distance.

elements introduced using indefinite NPs. I will call this the DISCOURSE COMPLEXITY ASSUMPTION.

4. If the current word is not compatible with the highest ranked structure built so far, there is reanalysis difficulty.

The storage and integration costs together provide a measure of INTEGRATION TIME, which gives the complexity at any given point, as well as overall complexity of a sentence. Gibson assumes the following regarding these two costs:

1. Integrations and storage access the same pool of resources.
2. Resources have fixed capacity.
3. Each predicted syntactic head takes up a fixed quantity of resources.
4. The overall acceptability of a sentence depends on the maximal integration time complexity experienced during the parsing process.

#### 2.3.4 Summary

Sentence processing according to the DLT is constrained by the limited resources in working memory. This limit on resources is quantified in terms of integration and storage costs of linguistic elements as they are processed in real time. There is a clear distinction between constraints on competence and on performance, and in this respect, Gibson's model differs significantly from Joshi's and Hawkins'. An attractive aspect of this model is the impressive array of cross-linguistic data that it can account for.

## 2.4 Lewis' Interference and Confusability Theory (2001)

The central thesis of the ICT is that syntactic similarity of items in a sentence is responsible for differences in processing difficulty. Syntactic similarity results in increased interference in human short-term or working memory, and this results in increased storage and retrieval difficulty. The details of the model are presented next, and are based on (Lewis & Nakayama, 2001).

As items (e.g., nouns, verbs) are processed in real time, they are incrementally attached to any syntactic structure that has been built so far, this syntactic structure being maintained in short-term or working memory. The attachment process is subject to one of two kinds of interference: PROACTIVE INTERFERENCE (PI) and RETROACTIVE INTERFERENCE (RI), which are discussed next.

### 2.4.1 Interference

PI and RI are defined as follows. PI occurs when a to-be-attached item is preceded by other items which have a similar syntactic status. For example, if a subject NP is to be attached, any other potential subjects preceding it will cause PI during the attachment. RI is the logical dual: the attachment of an item suffers from interference from other items which occur to its right and which have a similar syntactic status. For example, if a subject NP is to be attached, any other potential subjects following it will cause PI during the attachment.

In general and independent of parsing, interference in working memory can be characterized as RETRIEVAL INTERFERENCE or STORAGE INTERFERENCE. Retrieval interference is due to overlapping retrieval cues, while storage interference is the interference due to overlapping features of the items. In the context of sentence

parsing, when PI or RI occurs, the interference is retrieval interference if the interfering items have not already been attached to the existing structure built so far. If the interfering items have already been attached, the interference is storage interference. PI and RI can thus be a result of either retrieval interference or storage interference; the model assumes that when both PI and RI occur, they contribute equally to interference.

An example illustrates the working of the model. Consider a Japanese sentence like (8a); this can be schematically represented as (8b).

- (8) a. Ani-ga                    sensei-ni    onna-no-ko-ga asondeiru-to renrakusita  
elder brother-nom teacher-dat girl-nom        playing-that notified  
‘My older brother notified the teacher that a girl is playing.’
- b. NP1-ga NP2-ni NP3-ga V2 V1  
 $\phi$                                      $\rho$

In order to simplify the presentation, we begin at the point where V2 is being processed. The subject of V2 is the nominative case-marked NP3, and it suffers proactive retrieval interference from the as-yet-unattached NP1 ( $\phi$ ). By contrast, when V1 is being processed, its subject, NP1, suffers retroactive storage interference from the already attached NP3 ( $\rho$ ). In both cases, NP2 causes no interference since, unlike NP1 and NP3, it is not a potential subject.

Thus, parsing in ICT is driven by general constraints on human working memory, these constraints arising due to interference in the course of real-time processing. The total amount of interference at a given point is the sum of the two kinds of interference, and an increase in interference results in increased confusability between similar items. For ease of exposition, we assign simple integer values to each unit of interference; however, in the actual computational implementation, the costs are not necessarily simple integer values (Lewis, personal communication).

#### 2.4.2 Focus of attention and interference

The ICT also exists as a computationally implemented version cast in the cognitive modeling framework ACT-R (Anderson & Lebiere, 1998). Here, certain assumptions about general cognitive architecture additionally come into play. One important assumption, based on results from short-term memory research, is that PI occurs only when an item is out of the focus of attention; when an item is in the focus of attention, only RI applies. At any given time, two items are assumed to be in the focus of attention, the current one and the immediately preceding one. For example, in the Japanese sentence above, at V2 only RI applies, but if an adverb is inserted between the last NP and the innermost verb V2 as shown in (9), the final NP is out of the focus of attention, and therefore both RI and PI occur.

- (9) Ani-ga                    sensei-ni    onna-no-ko-ga asa        rokuji    kara  
    elder brother-nom teacher-dat girl-nom        morning 6-o'clock from  
    asondeiru-to renrakusita  
    playing-that notified  
    ‘My older brother notified the teacher that a girl has been playing since 6AM.’

This fact about the model results in an important empirical problem, as discussed in Section 2.5.

#### 2.4.3 Summary

The ICT proposes that syntactic similarity is the key factor affecting processing difficulty. Attractive features of this model are its simplicity and high degree of generality across cognitive tasks. RI and PI effects are attested in a wide variety of modalities in human cognition, and the specific assumptions about the syntactic

component of human working memory, e.g., positional similarity, have correlates in other domains (Lewis, 1996a).

## 2.5 Empirical problems with existing parsing models

All these models have trouble accounting for processing facts about center embeddings. I discuss the problems with each model next.

### 2.5.1 Joshi's EPDA

Consider again the Dutch vs. German contrast in example (3), repeated below:

- (10) a. Jan Piet Marie zag laten zwemmen  
Jan Piet Marie saw make swim  
'Jan saw Piet make Marie swim.'
- NP1 NP2 NP3 V1 V2 V3
- b. ...dass Hans Peter Marie schwimmen lassen sah  
...that Hans Peter Marie swim make saw  
'...that Hans saw Peter make Marie swim.'
- NP1 NP2 NP3 V3 V2 V1

Bach et al. showed that Dutch crossed dependencies are easier to process for Dutch native speakers than German nested dependencies for German native speakers. The EPDA is intended to model phrase-by-phrase processing difficulty (Joshi, p.c.), so it would predict that the highest processing cost is at the innermost verb in both the Dutch and German cases, since in the EPDA the most time is spent there and the number of items present in the EPDA at this point is the largest. However, experimental work has shown that this is not true, at least not for Dutch (Dickey & Vonk, 1997), (Kaan & Vasić, 2000), In Dutch, the most costly region seems to be

at the final NP. Moreover, in the EPDA, structure building does not begin until the verbs are reached; until that point, the NPs are simply stored in the stack. NPs, however, generate predictions (see, e.g., (Scheepers, Hemforth, & Konieczny, 1999), and references cited therein), they are not just merely stored in a temporary buffer in working memory. Thus, there are several empirical problems in the EPDA model: the inability to predict phrase-by-phrase reading times correctly for Dutch (there are no reading time studies for German self-center embeddings, as far as I know), and the assumption of simple storage of the NPs before the verbs are encountered.

In addition, the EPDA is unable to account for several other facts regarding Hindi center embeddings. The details are presented in (Vasishth, to appear, 2002); I summarize the main results below.

Like many other languages (Aissen, 2000) with no determiners such as English *the*, definiteness<sup>8</sup> in Hindi direct object noun phrases is marked by case marking, as the examples below illustrate.

- (11) a. Siitaa-ne Hari-ko [kitaab khariid-neko] kahaa  
       Sita-erg Hari-dat book buy-inf told  
       ‘Sita told Hari to buy a book.’
- b. Siitaa-ne Hari-ko [kitaab-ko khariid-neko] kahaa  
       Sita-erg Hari-dat book-acc buy-inf told  
       ‘Sita told Hari to buy the book.’

The parses for (11a,b) are identical, as shown in Figure 2.7.

From Figure 2.7 it is easy to see that the EPDA predicts the following for Hindi center embeddings: (i) No difference in processing difficulty at NP3 with respect

---

<sup>8</sup>Strictly speaking, the case marking marks specificity, which is a distinct notion semantically from definiteness. However, as discussed in (Vasishth & Joseph, 2002), definiteness is conversationally implicated in case marked direct object NPs presented out of context, unless an indefiniteness marker like *ek*, ‘one’, is also present.

Input head at	Stack sequence	Stack	Pop action	No. of items
NP1				0
NP2		NP1		1
NP3		NP1 NP2		2
V2		NP1 NP2 NP3		3
V1	V*2	NP1		4
	V*2		V1(NP1,S1)	3
			V2(NP2, NP3)=S1	0

Figure 2.7: Hindi examples (11a,b)

to definiteness marking; (ii) No difference in processing difficulty at the innermost verb in (11a) versus (11b); (iii) Greatest difficulty at final (matrix) verb. All these predictions turn out to be incorrect, as I show in (Vasishth, to appear, 2002).

(i) and (ii) above are not a particularly serious objection to Joshi’s model since the model makes no assumptions about the effect of discourse referents (such a theory can be incorporated into the model). But (iii) *is* a serious problem for the model.

### 2.5.2 Hawkins’ EIC

Hawkins argues that the EIC is exclusively responsible for constraints on processing, and the apparent effect of information structure (i.e., given versus new information) is actually due to the differing lengths of constituents (short constituents signaling old information and long ones signaling new information) (Hawkins, 1998, 751-759).

This view predicts that merely appending the definiteness marker *-ko* on a direct object in Hindi should have no effect of processing difficulty. For example, the two sentences in (12) should show no difference in processing difficulty, since the direct object *kitaab*, ‘book’ stands in the same structural relationship to the other elements. It is irrelevant for EIC that it is an indefinite NP in one and a definite NP in the

other.

- (12) a. Siitaa-ne Hari-ko [kitaab khariid-neko] kahaa  
Sita-erg Hari-dat book buy-inf told  
'Sita told Hari to buy a book.'
- b. Siitaa-ne Hari-ko [kitaab-ko khariid-neko] kahaa  
Sita-erg Hari-dat book-acc buy-inf told  
'Sita told Hari to buy the book.'

A second prediction relates to scrambled sentences, in which IC-to-word ratios change. The EIC metric suggests that: (i) there should be no difference between (13a) and (14a) below; and (ii) definiteness marking on the direct object should have no effect on processing difficulty, as discussed above (compare (13b) and (14b) with (13a) and (14a), respectively).

- (13) a. Hari<sub>i</sub>-ko Siitaa-ne *t<sub>i</sub>* [PRO<sub>i</sub> kitaab khariid-neko] kahaa  
Hari-dat Sita-erg book buy-inf told  
(Lit.) 'It was to Hari that Sita said to buy a/the book.'
- b. Hari<sub>i</sub>-ko Siitaa-ne *t<sub>i</sub>* [PRO<sub>i</sub> kitaab-ko khariid-neko] kahaa  
Hari-dat Sita-erg book-acc buy-inf told  
(Lit.) 'It was to Hari that Sita said to buy the book.'
- (14) a. kitaab<sub>j</sub> Siitaa-ne Hari<sub>i</sub>-ko [PRO<sub>i</sub> *t<sub>j</sub>* khariid-neko] kahii  
book-fem Sita-erg Hari-dat buy-inf told-fem  
(Lit.) 'As for the book, Sita told Hari to buy it,'
- b. kitaab<sub>j</sub>-ko Siitaa-ne Hari<sub>i</sub>-ko [PRO<sub>i</sub> *t<sub>j</sub>* khariid-neko] kahaa  
book-fem-acc Sita-erg Hari-dat buy-inf told  
(Lit.) 'As for the book, Sita told Hari to buy it,'

Finally, EIC also predicts that inserting an adverb between the final NP and the first verb should result in a lower IC-to-word ratio. That is, examples like (15a) should be significantly easier to process than (15b). Specifically, if EIC applies in

real-time processing, it should take longer to process the innermost verb in (15b) than in (15a), since the IC-to-word ratio is lower in (15b).

- (15) a. Siitaa-ne Hari-ko [kitaab-ko khariid-neko] kahaa  
 Sita-erg Hari-dat book-acc buy-inf told  
 ‘Sita told Hari to buy a/the book.’
- b. Siitaa-ne Hari-ko [kitaab-ko **jitnii-jaldi-ho-sake** khariid-neko] kahaa  
 Sita-erg Hari-dat book-acc as-soon-as-possible buy-inf told  
 ‘Sita told Hari to buy the book as soon as possible.’

However, the experimental results presented in (Vasishth, to appear, 2002) directly contradict these predictions: definiteness marking results in *increased* processing difficulty, scrambling indirect objects has a very different effect on processing difficulty compared to the scrambling of direct objects (with definiteness marking adding another dimension of complexity), and inserting an adverb between the final NP and the innermost verb results in *decreased* processing difficulty at the verb. In other words, the EIC alone cannot account for processing difficulty in Hindi.

### 2.5.3 Gibson’s DLT

One interesting prediction of the DLT for Hindi center embeddings is that integration cost at the innermost verb increases monotonically as a function of increasing integration distance. For example, consider the single center-embeddings in (16).<sup>9</sup>

- (16) Siitaa-ne Hari<sub>i</sub>-ko [PRO<sub>i</sub> kitaab khariid-neko] kahaa  
 Sita-erg Hari-dat book buy-inf told  
 ‘Sita told Hari to buy a/the book.’

---

<sup>9</sup>Hindi center embeddings are obligatory control constructions (Bickel & Yadava, 2000), (Vasishth, to appear, 2002): the indirect object is coindexed with the PRO in the embedded clause. In the DLT, null elements play no part in computing storage costs, but are relevant for computing integration costs, as shown in the figures.

Figure 2.8 summarizes the storage and integration costs, and Figure 2.9 shows how the integration costs are computed.

Input word:	Siitaa-ne	Hari-ko	kitaab	khariid-neko	kahaa
Storage cost (MUs):	2	1	1	1	0
Integration cost (EUs):				1	

Figure 2.8: Integration and storage costs for single center embeddings

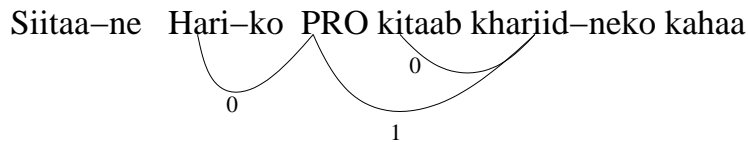


Figure 2.9: Computation of integration costs at innermost verb

Integration distance can be increased by dislocating the indirect object or the direct object to sentence-initial position; as I show below, the greater increase is due to direct-object dislocation. In addition, if we include optional definiteness marking (*-ko*) on direct objects, the various ordering possibilities can be summarized as shown in (17) and (18). In each case, I mark the original position of the dislocated element with a coindexed trace  $t_i$ ; this is merely for expository purposes, and does not imply any assumptions regarding the existence of traces.

- (17) a. Hari<sub>*i*</sub>-ko Siitaa-ne  $t_i$  [PRO<sub>*i*</sub> kitaab khariid-neko] kahaa  
 Hari-dat Sita-erg book buy-inf told  
 (Lit.) ‘It was to Hari that Sita said to buy a/the book.’
- b. Hari<sub>*i*</sub>-ko Siitaa-ne  $t_i$  [PRO<sub>*i*</sub> kitaab-ko khariid-neko] kahaa  
 Hari-dat Sita-erg book-acc buy-inf told  
 (Lit.) ‘It was to Hari that Sita said to buy the book.’

- (18) a.  $kitaab_i$  Siitaa-ne Hari $_j$ -ko [PRO $_j$   $t_i$  khariid-neko] kahii  
 book-fem. Sita-erg Hari-dat buy-inf told-fem  
 (Lit.) ‘As for the book, Sita told Hari to buy it,’
- b.  $kitaab_i$ -ko Siitaa-ne Hari $_j$ -ko [PRO $_j$   $t_i$  khariid-neko] kahaa  
 book-fem-acc Sita-erg Hari-dat buy-inf told  
 (Lit.) ‘As for the book, Sita told Hari to buy it,’

The Figures 2.10-2.13 below summarize the procedures for computing storage and integration costs for examples (17a) and (18a); the computations for (17b) and (18b) are the same, except that there is decreased processing cost at definite (case-marked) direct objects due to the discourse complexity assumption (see Section 2.3.3).

The integration and storage cost computations for indirect object dislocation are shown below in Figures 2.10 and 2.11, and those for direct object dislocation are shown in Figures 2.12 and 2.13.

Input word:	Hari-ko	Siitaa-ne	kitaab	khariid-neko	kahaa
Storage cost (MUs):	1	1	1	1	0
Integration cost (EUs):				2	

Figure 2.10: Integration and storage costs for IO-dislocated center embeddings (example (17))

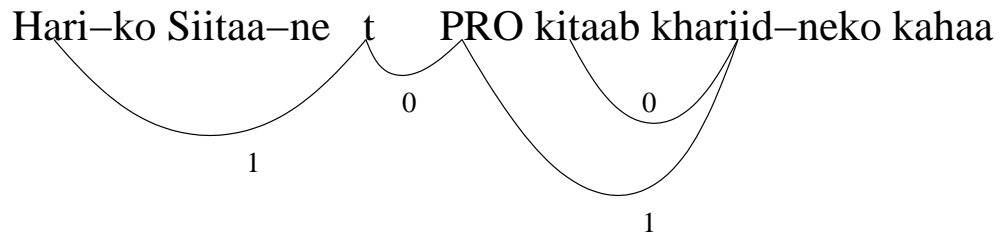


Figure 2.11: Integration costs at innermost verb for IO dislocation (example (17))

The DLT also predicts that increasing distance by inserting an adverb will not affect integration cost (this is because an adverb does not introduce a discourse marker). Consider the examples in (19).

Input word:	kitaab	Siitaa-ne	Hari-ko	khariid-neko	kahii
Storage cost (MUs):	1	1	1	1	0
Integration cost (EUs):				3	

Figure 2.12: Integration and storage costs for DO-dislocated center embeddings (example (18))

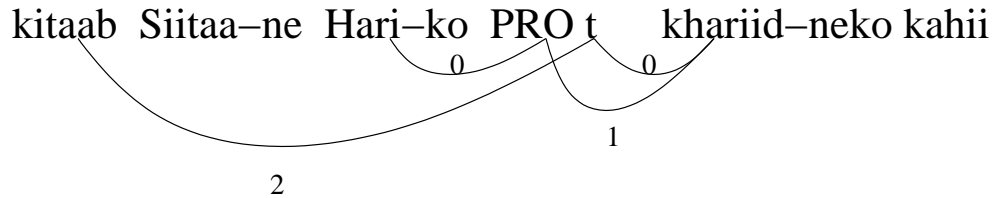


Figure 2.13: Integration costs at innermost verb for DO dislocation (example (18))

- (19) a. Siitaa-ne Hari-ko [kitaab-ko khariid-neko] kahaa  
 Sita-erg Hari-dat book-acc buy-inf told  
 ‘Sita told Hari to buy a/the book.’
- b. Siitaa-ne Hari-ko [kitaab-ko **jitnii-jaldi-ho-sake** khariid-neko] kahaa  
 Sita-erg Hari-dat book-acc as-soon-as-possible buy-inf told  
 ‘Sita told Hari to buy the book as soon as possible.’

Processing cost remains unchanged in the DLT irrespective of the presence or absence of the adverb. Note, however, that Gibson allows the possibility of non-discourse marker elements introducing processing difficulty; if we admit this possibility, the DLT would predict an increase in integration cost at the innermost verb.

Input word:	Siitaa-ne	Hari-ko	kitaab (adv.)	khariid-neko	kahaa
Storage cost (MUs):	2	1	1	1	0
Integration cost (EUs):				1	

Figure 2.14: Integration and storage costs for center embeddings with or without an adverb

Finally, the DLT predicts that indefinites will be harder to process than definites: The discourse complexity assumption (see Section 2.3.3) predicts that processing load will be greater at the direct object *kitaab*, ‘a book’, compared to *kitaab-ko*, ‘the book’.

I summarize the DLT’s predictions below:

1. The greatest processing difficulty is at the innermost verb.
2. Integration cost at innermost verb is greatest in IO-dislocated sentences, and least in canonical-order sentences.
3. Adverbs could increase integration cost.
4. Indefinites are harder to process than definites.

As discussed in (Vasishth, to appear, 2002), all these predictions turn out to be incorrect: the greatest processing load is at the prefinal or final NP, and there is consistently a speedup in processing at the verb (this is consistent with the Dutch results (Dickey & Vonk, 1997), (Kaan & Vasić, 2000)); no difference was found in processing difficulty at the innermost verb for indirect- versus direct-object dislocated NP sentences and (as mentioned earlier) inserting an adverb results in *decreased* reading time at the innermost verb; and indefinites are *easier* to process than definites. In sum, the complexity metric specified by the DLT is unable to account for the Hindi processing facts.

#### 2.5.4 Lewis’ ICT

An important (alleged) problem for Lewis’ ICT is pointed out in (Gibson, 2000). (Warren & Gibson, 1999) have shown that sentences with embedded first- or second-

person pronouns (such as the pronoun *I* in (20a)) are rated to be significantly easier to comprehend than sentences with proper names or definite descriptions, as in (20b,c).

- (20) a. The reporter [who the senator [who I met] attacked] disliked the editor.  
b. The reporter [who the senator [who John met] attacked] disliked the editor.  
c. The reporter [who the senator [who the professor met] attacked] disliked the editor.

The problem for the ICT is that the syntactic positions of the underlined words are identical, and so no difference in processing difficulty is predicted in sentences (20a,b,c). Experimental results such as these led Gibson to hypothesize that discourse-new referents are harder to process than presupposed or old referents. In Gibson's model, first and second-person pronouns are assumed to introduce no discourse referents, and this leads to the DLT correctly predicting the observed difference.

However, apart from the fact that discourse-referent effects are not inconsistent with the ICT, it is not clear whether the above result for English is cross-linguistically valid. (Kaan & Vasić, 2000) found that in Dutch comprehension difficulty increases with pronouns compared to proper names, and (Vasishth, to appear, 2002) has shown that in Hindi definite descriptions result in greater processing difficulty compared to indefinites, results that are the opposite to that discussed for English in (Warren & Gibson, 1999). The Dutch and Hindi results suggest that Gibson's alternative formulation of processing difficulty in terms of discourse referents may not be valid cross-linguistically.

Regarding Hindi center embeddings, the ICT makes the correct predictions for the most part. One problem for the ICT (in its current formulation) relates to the

effect of adverb insertion discussed earlier in the context of Joshi's, Hawkins', and Gibson's models.

As discussed in (Vasishth, to appear, 2002), the ICT incorrectly predicts that inserting an adverb between the final NP and the first verb should result in greater processing difficulty at the innermost verb. That is, examples like (21a) should be significantly easier to process than (21b). This difference is predicted because ICT assumes that items that have been out of the focus of attention for a given period of time suffer increased proactive interference

- (21) a. Siitaa-ne Hari-ko [kitaab-ko khariid-neko] kahaa  
Sita-erg Hari-dat book-acc buy-inf told  
'Sita told Hari to buy a/the book.'
- b. Siitaa-ne Hari-ko [kitaab-ko **jitnii-jaldi-ho-sake** khariid-neko] kahaa  
Sita-erg Hari-dat book-acc as-soon-as-possible buy-inf told  
'Sita told Hari to buy the book as soon as possible.'

As discussed earlier, the opposite result is observed in Hindi.

In sum, although there is much cross-linguistic evidence supporting all the four models, many facts about center embeddings remain difficult to account for. In the next chapter I present a model that can overcome these empirical problems.

## CHAPTER 3

### HUMAN SENTENCE PARSING AS ABDUCTION

In a very general way, it is clear what kind of models we need, because one basic feature of the organization of social insect colonies is already obvious. Individuals, following simple, local rules, generate the achievements of the colonies.

*Ants at Work*, by Deborah Gordon, p. 143

#### 3.1 Introduction

Humans routinely engage in automatic or deliberative reasoning in order to process information about the world around them. In the course of almost any information processing task, we develop one or more hypotheses that “best explain” the facts, and either verify or discard them as further evidence emerges. This general cognitive mechanism is evident in diverse activities like diagnosing illness, solving crimes, developing scientific theories, visual processing, and comprehending language. Peirce was the first to discuss this third kind of reasoning process (distinct from deduction and induction); he called it ABDUCTION (see (Josephson & Josephson, 1996) and the references cited therein for a detailed discussion). Abduction is the following pattern of reasoning, and often is characterized as an “inference to the best explanation” (Harman, 1965).

$D$  is a collection of information available to us

Hypothesis  $h$  explains  $D$

No other hypothesis explains  $D$  as well as  $h$

Therefore,  $h$  is probably correct.

Abductive inference has been proposed as an explanatory mechanism for language perception and processing; to give a few examples: text comprehension (Smith, 2000), text generation (Klabunde & Jansche, 1998), semantic processing (Strigin, 1998), (Strigin, 2001), pragmatic interpretation (Hobbs, Stickel, Appelt, & Martin, 1993), and (chart parser-based) grammar-rule acquisition/learning (Cussens & Pulman, 2000). In this thesis, I propose that sentence comprehension difficulty results from a series of automatic abductive inferences. I also show that, when human parsing is viewed from this perspective, phrase-by-phrase processing difficulty in center embeddings can be accounted for with greater accuracy than is possible with other models.

### 3.1.1 An informal outline of the model

The main intuition behind the proposed model is that one or more hypotheses regarding the structure of the sentence are generated as each word in a sentence is processed; the hypothesis-generation process is constrained by a principle called MINIMAL CONSISTENCY (an application of Ockham's razor), which says that no words of a certain syntactic category will be predicted if there is no evidence for them. These hypotheses are assumed to be stored in human working memory and overload working memory capacity under certain well-defined conditions (discussed later).

### 3.1.2 Granularity versus breadth of coverage: Some caveats

As mentioned in the introductory chapter, the model in its present form aims to account only for processing complexity associated with center-embedding constructions. In addition, the discussion is limited to Dutch, German, Hindi, Japanese, and Korean, i.e., languages which have the word order (at least descriptively) subject-object-verb, and in which center embedding constructions have been observed to occur naturally (e.g., in text corpora). I limit the scope of this thesis in this manner because, as discussed in Section 1.2, page 4, even for such a highly constrained set of languages and constructions, currently there are no cross-linguistically satisfactory accounts.

For the same reason, the present model does not attempt to provide a broad-coverage parsing model. Choosing between such “toy” models and broad-coverage models presents a tradeoff between granularity and breadth of coverage. On the one hand, models such as Joshi’s, Gibson’s, and Lewis’ try to predict phrase-by-phrase processing difficulty, but it is unclear whether these would scale up to “psychologically real” parsers for natural language applications. On the other hand, parsers proposed by Jurafsky (1996), Roark (2001), and Brants and Crocker (2000) aim for broad coverage, but it is unclear whether they can predict phrase-by-phrase processing difficulty in general, i.e., beyond predictions of garden-path effects. Hawkins’ model appears to lie between these two extremes: it aims to account for both low-level processing difficulties and has broad coverage. However, it also is limited in its ability to correctly predict processing difficulty at a fine level of granularity (Konieczny, 2000), (Vasishth, to appear, 2002).

Of course, both kinds of goals, granularity and broad coverage, are worth pursuing. However, the former primarily seeks to identify constraints on human

cognition at a low level, while the latter seeks to cover a broader class of phenomena at the expense of explanatory detail. In this thesis, I follow Joshi, Gibson, and Lewis, and limit myself to addressing the granularity problem. A long-term goal is to build a broad-coverage and scalable system and recent research (Narayanan & Jurafsky, 1998), (Briscoe, 2000), (Hale, 2001) has already suggested ways to do this.<sup>1</sup>

## 3.2 Processing as abduction+deduction: The main proposal

### 3.2.1 The underlying competence grammar $\mathcal{G}$

I assume a grammar  $\mathcal{G}$  for a particular natural language  $\mathcal{L}$ .  $\mathcal{G}$  defines what types of functional categories (or predicate-valency frames) we can encounter in  $\mathcal{L}$ , and how these functions can combine with their arguments. Throughout, I assume that  $\mathcal{G}$  is a categorial grammar along the lines of (Kruijff, 2001). In a lexicalist approach like categorial grammar, the lexicon determines how words can be put together. Structural rules, like those in Multi-Modal Logical Grammar (MMLG) (Moortgat, 1997), only vary the order in which words occur grammatically.<sup>2</sup> The grammar system assumed here is not necessarily limited to the dependency grammar based framework of Kruijff; it could in principle be a system along the lines of an appropriate formalization of Chomskyan minimalism (Vermaat, 1999), (Retoré & Stabler, to appear), Tree-adjointing grammar (Abeillé & Rambow, 2000), Combinatorial Categorial Grammar (Steedman, 2000), or Head-driven Phrase Structural Grammar (Pollard & Sag, 1994).

---

<sup>1</sup>Narayanan and Jurafsky, and Hale present variants of probabilistic Earley parsers along the lines of (Stolcke, 1995), while Briscoe presents a deterministic shift-reduce parser. Briscoe's simulations, even though they relate to artificial languages, are particularly interesting and relevant in the context of the present model, but it would take us too far afield to discuss these in any detail here.

<sup>2</sup>Moortgat calls the system Multimodal Categorial Grammar; here, I follow the terminology used in Kruijff (2001).

Any choice of grammar system will carry its own implications; I leave the exploration of these options to future research.

### 3.2.2 The set $\mathcal{H}$ of hypothesis formers

A set of HYPOTHESIS FORMERS  $\mathcal{H}$  is created on the basis of a compilation of the lexicon.<sup>3</sup> This compilation is based on a procedure proposed for linear logic in (Hepple, 1998), and extended in (Kruijff, 1999) to cover a larger range of the multiplicative resource logics used in MMLG.<sup>4</sup> The result of the procedure is a set of first-order functions to represent categories (i.e., there are no higher-order formulas).<sup>5</sup>

$\mathcal{H}$  essentially provides us with schemas that encode information about how words (of particular categories) can be combined. For example, intransitive verbs give us the schema  $f(NP1)$ , transitive verbs  $f(NP1, NP2)$ , and so on. These hypotheses can be considered to be tree structures such as those shown in Figure 3.1, but for typographical convenience I will use the functional notation.



Figure 3.1: Tree representation of schemas

---

<sup>3</sup>Compilation here does not mean the creation of a lexicon; rather, given a lexicon, a set of procedures are applied to it in order to compile out information present in the lexicon.

<sup>4</sup>Compilation was originally proposed for the purposes of efficient chart parsing with Lambek-style grammars, in order to overcome problems with earlier approaches, e.g., (Hepple, 1992) and (König, 1990).

<sup>5</sup>See (Console, Portinale, & Dupré, 1996) for a similar approach for rendering diagnosis more efficient.

I assume that  $\mathcal{H}$  is finite and closed.<sup>6</sup> These are all reasonable assumptions since  $\mathcal{H}$  cannot have any redundant hypothesis formers (having been created from the grammar rules), and the schemas extracted from the grammar will be finite (if this were not so, the set of grammar rules would be infinite).

The set of hypothesis formers  $\mathcal{H}$  is assumed to be partially ordered by a SIMPLICITY CRITERION: simpler schemas appear before the more complex ones. For example, monoclausal schemas are simpler than biclausal ones, the intransitive verb schema is simpler than the ditransitive verb schema. This assumption is based on experimental evidence from (Yamashita, 1997), which showed that (Japanese) subjects prefer to complete sentences with verbs that are simpler (i.e., verbs that result in monoclausal structures) rather than more complex ones. I take this result to suggest that simpler structures are accessed before more complex ones, and model this assumption by the partial ordering (the ordering is partial because it is possible that there is no way to specify relative simplicity between a given pair of hypotheses).

The precise ordering criteria may vary from language to language and even within a language, and it is also possible that a more sophisticated set of decision criteria are needed (such as the frequency of certain syntactic structures) in order to determine the ordering. As an example of different results in two unrelated languages, consider Japanese and Hindi. For Japanese, Uehara has found (M. Nakayama, personal communication) that (contra Yamashita's findings above) Japanese subjects prefer two clauses over one in incomplete sentences beginning with two nominative-case marked NPs; in Japanese, this sequence could be continued either with a stative verb or a bi-clausal structure. The simplicity criterion (see above) wrongly predicts that the stative verb (monoclausal structure) is ordered before the biclausal one. However,

---

<sup>6</sup>This is not to be confused with the fact that the set of sentences in  $\mathcal{L}$  is infinite.

research on Hindi gives a different result: (a) in sentence completion tasks involving sentences beginning with three NPs, subjects prefer to use ditransitive verbs rather than transitive one (which would entail a biclausal structure), and (b) in a sentence comprehension study, we find that the reading time in sentences beginning with four NPs is shorter for ditransitive embedded verbs than that for transitive embedded verbs (Vasishth, to appear, 2002).

### 3.2.3 Some definitions

Next, I define some terms that I use in the proposed algorithm.

#### **Abducible structure(s):**

An ABDUCIBLE STRUCTURE is a schema/hypothesis that can be abduced based on (a) the principle of minimal consistency, and (b) the information available so far.

New information results in previous hypotheses being replaced. Abduced *functions*  $f_i$  are part of the abducible structures that are taken from  $\mathcal{H}$ , and thus predict the presence of a word with a particular syntactic category. For example, in Japanese, if only a nominative NP (we represent this as  $NP[nom]$ ) has appeared so far,  $f_i(NP[nom])$  is an abducible structure that says: an intransitive verb  $f_i$  with the nominative NP will give a sentence.<sup>7</sup>

Although a nominative case marked NP is in principle consistent with an infinite set of possible continuations, our model allows for the selection of only those

---

<sup>7</sup>The subscript on  $f$  is merely a notational device used in the derivations to distinguish one abduced function from another.

hypotheses from the hypothesis formers  $\mathcal{H}$  that are MINIMALLY CONSISTENT with the nominative NP. Minimal consistency is defined as follows:

**Minimal consistency:**

There are two cases: (i) the current word<sup>8</sup> is an NP, and (ii) the current word is a verb.

(i) If the current word is an NP: A list<sup>9</sup> of hypotheses  $H \subset \mathcal{H}$ , is minimally consistent with the current list of words iff each hypothesis  $h \in H$  can instantiate the NPs and verbs seen so far (in the current list of words) as arguments and functions respectively, without positing any new, unseen arguments.

(ii) If the current word is a verb: A list of hypotheses  $H \subset \mathcal{H}$ , is minimally consistent with a verb iff each hypothesis  $h \in H$  satisfies one of the following two conditions:

Subcase 1: The hypothesis  $h$  is able to take all the preceding words as arguments and/or functions and can MATCH (matching is defined below) the current verb with a function in  $h$ .

Subcase 2: If the matching process fails, the current verb becomes part of a hypothesis  $h$  which saturates<sup>10</sup> all the arguments and verbs seen previously. If the current verb requires any new, unseen argument(s) and/or is necessarily an argument of another as-yet-unseen function  $f_i$ , the unseen argument(s)  $n$  and/or the

---

<sup>8</sup>For simplicity, I treat a noun phrase (NP) as consisting of a single word.

<sup>9</sup>This is actually a set, ordered by the simplicity criterion (see page 44).

<sup>10</sup>Saturation is to be understood in the standard linguistic sense of a verb instantiating all of its arguments.

function  $f_i$  are posited. Any unseen arguments that the function  $f_i$  would then require must also be present in each hypothesis  $h$ .

An example illustrating the first clause above of minimal consistency is as follows. Suppose that, during the course of processing a Japanese sentence, we have seen only one nominative NP so far. In that case, a hypothesis satisfying minimal consistency is  $f_i(NP[nom])$ , and one *violating* minimal consistency is:  $f_i(NP[nom], n)$ , where  $n$  is a hypothesized, new, unseen NP. By contrast, if, after we see the first nominative NP, we see a second nominative NP, the minimally consistent hypotheses are now  $f_i(NP1[nom], NP2[nom])$ , where  $f_i$  is a stative verb, and  $f_{i+1}(NP1[nom], f_{i+2}(NP2[nom]))$ , i.e., a center-embedded structure. This is because Japanese allows only these two structures with two nominative case-marked NPs.

Subcase 1 of the second clause, in which a verb is seen after some NPs, requires a process of matching the verb to a hypothesized function; matching is defined as follows.

**Matching:** A verb  $V$  MATCHES with a function  $f_i$  iff  $V$  has a valency that is identical with that of  $f_i$ , and all the arguments in the subcategorization frame of the verb MATCH those of  $f_i$ .

An NP can MATCH with a posited NP argument iff its case marking, person, number, gender, and other information, is subsumed by (informally, “is consistent with”; see (Shieber, 1986)) that of the posited argument.

I illustrate this with the Japanese example above. If, after seeing two nominative case marked NPs, one sees an intransitive embedded verb, this verb is matched successively

with the functions  $f_i$  to  $f_{i+2}$  in the abduced hypotheses,  $f_i(NP1[nom], NP2[nom])$ , and  $f_{i+1}(NP1[nom], f_{i+2}(NP2[nom]))$ . The match succeeds with  $f_{i+2}$ .

Subcase 2 of the second clause involves a situation where matching has failed; it can be exemplified as follows. Suppose that the Hindi sentence in (22) is being processed. The first item that we see is a verb V1, *kahaa*, ‘said[past].’ At this point, the hypothesis will be  $V1(n1, n2, f_i(n2))$ , because V1 necessarily has a subject  $n1$  and an object  $n2$ ; in addition, since V1 is an obligatory subject control verb (Vasishth, to appear, 2002), an embedded clause headed by a function (verb)  $f_i$  must be posited, and its subject is  $n2$ .

- (22) *kahaa Siitaa-ne Hari-ko so-neko*  
 said Sita-erg Hari-dat sleep-inf  
 ‘Sita told Hari to (go to) sleep’.

The minimal consistency constraint is motivated by the production study by Yamashita (1997), and the pilot production study and the comprehension studies for Hindi by Vasishth (to appear, 2002) that were mentioned earlier.

With these definitions in place, we turn next to the algorithm, based on which the complexity metric is defined.

### 3.2.4 The algorithm

The top level of the processing algorithm is a Scan-Lookup-Process loop that terminates when there is no more remaining input. **Scan**() is a function that returns the next word  $w$  to be processed, **Lookup**( $w$ ) is a function that returns the lexical entry  $L_w$  of word  $w$ , and **Process**( $S$ ) is a function that carries out the abductive or deductive step using  $S$ , a list containing the lexical entries of the input so far. The

results of **Process** affect the contents of a list data structure,  $M$ (emory);  $M$  models the role of working memory.

```

S ← []      (Initialized to empty list)
M ← []      (Initialized to empty list)
while remaining input nonnull do
  w ← Scan()
  Lw ← Lookup(w)
  S ← append(S,Lw)
  Process(S)
end while

```

**Algorithm 1:** Main loop of the parser

```

Process(S)
if lastitem(S)==nominal then
  if ∃ abduced nominals in M then
    match(lastitem(S),M)
  else
    M ← Abduce(S, H)
  end if
else if lastitem(S)==verbal then
  if ∃ abduced function in M then
    match(lastitem(S),M)
  else
    M ← FindHyp(S, H)
  end if
end if

```

**Algorithm 2:** The main function **Process**(S)

**Abduce** returns all hypotheses  $h \in \mathcal{H}$  such that  $S$  and  $h$  are minimally consistent as defined earlier. **FindHyp** searches the set  $\mathcal{H}$  of hypothesis-formers for a hypothesis  $h$  that is a schema for the latest item in  $S$  (a verb).

To repeat the earlier example from Japanese: two nominative case marked NPs starting a sentence could be followed either by a stative predicate (23a), or a nested dependency construction with a single level of embedding (23b).

(23) a.  $f_2(NP1[nom], NP2[nom])$

$$b. f_3(NP1[nom], f_4(NP2[nom]))$$

These are two hypotheses selected from  $\mathcal{H}$ . No other hypotheses are selected because these are the only two that are minimally consistent, given the information so far. These hypotheses are based on the grammatical possibilities in Japanese, and since a single clause sentence has a simpler structure than a sentence with an embedded clause, the hypotheses are ordered as shown above. Next, the appearance of an accusative case marked NP will result in these hypotheses being discarded and the new hypothesis shown in (24) being selected:

$$(24) f_5(NP1[nom], f_6(NP2[nom], NP3[acc]))$$

Since the number of hypotheses decreases from two to one, the model predicts faster processing at the accusative NP. This prediction is borne out, as discussed later. We turn next to the complexity metric.

### 3.2.5 The complexity metric

The complexity metric has two components: **ABDUCTION COST**, the cost associated with the abductive process, and **MISMATCH COST**, the cost associated with a mismatch between an encountered verb and abduced functions.

**Abduction cost:** This reflects the increasing processing load as sentence fragments appear incrementally. The abduction cost is the sum of the number of NPs seen so far, the number of functions  $f_i$  that are posited (in all the the current abduced hypotheses), and the total number of current distinct hypotheses. These three sub-components are intended to reflect the load in working memory of: (a) storing an increasing number of NPs; (b) positing functions; and (c) storing hypotheses.

**Mismatch cost:** I assume that the (queued) hypotheses initially are unanalyzed units. By the term “unanalyzed units” I simply mean that when a hypothesis like  $f_i(NP1, f_j(NP2))$  is present in working memory and a verb is encountered, any attempt to match the verb with any of the functions  $f$  present in the hypothesis must be a left to right depth first search; the verb cannot directly match the right function. During this search process, every time a verb fails to match with a hypothesized function, there is a mismatch cost of one. Another way of saying this is that matching must be preceded by an unpackaging of the hypothesis in order to access the possible candidates for a match in the hypothesis. This unpackaging has a cost associated with it, which I quantify in terms of mismatch cost. The use of mismatch cost is merely a simplifying assumption; the computation of the unpackaging cost could equally be linked to some other plausible measure of complexity.

The hypotheses are assumed to be unanalyzed units on the basis of considerable research in psychology suggesting that humans have difficulty disengaging components of items (such as words and visual patterns) in memory (Johnson, 1977), (Johnson, 1981), (Johnson, Turner-Lyga, & Pettergrew, 1986), (Johnson, 1991). Of particular relevance is Johnson’s pattern-unit model, whose core assumption (Johnson, 1991, 83) is that precognitive representations of small visual patterns are always processed by first attempting to assign a single pattern-level cognitive representation of the array, and that step always succeeds if participants know the rule system or have a prelearned code that can be used for the assignment. The holistic encoding or representation is functionally motivated: the internal components of a pattern are functionally unnecessary for *storing* the pattern. As Johnson (1991, 84) puts it: “If participants would need to identify a component of an array, they always would be delayed in doing so by the initial attempts to encode the pattern as a whole.” The

pattern-unit model has considerable support from visual and language processing experiments (see the references cited above), and is the basis for assuming an initially holistic representation of hypotheses during the stage of encoding in working memory.

The numerical value associated with each sub-component in the metric is assumed to be 1, and the components are assumed to be additive. This is merely a convenience, and nothing crucial hinges on this assumption. In a fully implemented version of this model, the unit costs associated with each component would need to be correlated with precise reading time predictions.

The complexity metric depends crucially on the control structure of the parser: at each stage when the parser incrementally builds/revises the list of possible hypotheses, the complexity metric is used to compute the processing cost at that point. This is significant because it is well known that left-corner parsing, as opposed to pure top-down parsing, or head corner parsing, etc., is exactly the right strategy for differentiating the processing difficulty associated with center embeddings, left-branching, and right-branching structures (see Appendix 1).

### 3.3 Computational properties of the abductive parser

The proposed parser is a variant of the shift-reduce parser, LR(0), the LR parser with no lookahead (Aho & Ullman, 1993), (Sikkel, 1997). That is, the parser has a hybrid bottom-up/top-down discipline for building hypotheses. Although the process of hypothesis generation may appear to be indistinguishable from prediction (the top-down step(s) in the LR parser), prediction is a consequence of the generation of explanatory hypotheses. It is due to this distinction that I refer to the model as an *abductive* inference based model.

Unlike existing LR parsing algorithms, however, the present parsing model builds predictions incrementally as subsequences successively become available. These predictions are not all possible completions of the subsequence (which would be an infinite set), but only the ones satisfying minimal consistency. For an input string of length  $n$ , and given a set of hypothesis formers  $\mathcal{H}$ , where  $|\mathcal{H}| = k$ , parsing will take at most  $n \times k$  time, since each new item in the string will require searching the set  $\mathcal{H}$  for hypotheses.<sup>11</sup> Since  $k$  is constant for a given language, the worst-case time complexity of the algorithm is  $O(n)$ . Worst-case space complexity is as for left-corner parsers (see Appendix 1), but in the present model space complexity does not correspond directly to processing difficulty. Instead, the memory cost computation algorithm determines moment-by-moment difficulty.<sup>12</sup>

The present parsing model also raises a question heatedly debated in the literature: does the human parser require nondeterministic parsing? Although works like (Marcus, 1980), (Shieber, 1983), (Briscoe, 1987), (Briscoe, 2000) suggest that a nondeterministic parser is unnecessary for natural language, they all face problems when we evaluate such parsers with respect to experimental data. For example, Marcus’ parser employs a lookahead for making parsing decisions, and Shieber’s parser employs a related notion called “preterminal delay”.<sup>13</sup> This particular brand of bounded nondeterminism actually goes against the fact that humans process sentences strictly incrementally (Tyler & Marslen-Wilson, 1977). Briscoe (1987) believes that

---

<sup>11</sup>It could take less time if the set  $\mathcal{H}$  of hypothesis formers is not assumed to be a flat data structure but is structured as a hierarchy of hypotheses, so that adopting one hypothesis rules out entire subclasses of hypotheses as possible candidates. This detail does not affect the model presented here.

<sup>12</sup>Note that worst-case time complexity results are not directly useful; much more useful is average-case time complexity. However, as far as I know, there are no mathematically determined average-case results for left-corner parsers, only empirically determined ones (Joshi, 1996, 407).

<sup>13</sup>Interestingly, even Shieber’s deterministic shift-reduce parser is actually “a simulation of a nondeterministic machine” (Shieber, 1983, 701).

other cues, such as prosody, always disambiguate sentences, but this cannot account for the center embedding processing facts summarized in Chapter 2. Furthermore, it is not clear how the nondeterminism associated with dislocated constituents in English can be disambiguated with prosody. These are sentences such as *Mary<sub>i</sub>, I like to tell jokes to e<sub>i</sub>*. Here, the gap for the dislocated element *Mary* cannot be identified until the very end of the string is reached (Johnson-Laird, 1983).

The present model differs from these other approaches to bounded nondeterminism in the following way. More than one hypothesis can be generated by the parser, depending on the grammar of a particular language. The model is thus technically a nondeterministic pushdown automaton, and it simulates nondeterminism in a manner quite similar to “preterminal delay” in (Shieber, 1983). However, the difference is that the present model allows incremental interpretation at every stage in the processing; this is possible because the hypotheses that are generated contain all the information needed to build partial semantic interpretations. Thus, the parser ends up being equivalent to a deterministic pushdown automaton. This becomes relevant when we consider the problems associated with Joshi’s embedded pushdown automaton account, and possible revisions of the EPDA (see Section 6.1).

This concludes the presentation of the proposed model. The empirical coverage of the abductive inference based processing model is presented in the next chapter.

## CHAPTER 4

### EMPIRICAL COVERAGE OF THE PARSING MODEL

Model-making goes along with empirical work that investigates whether the kinds of behavior suggested by the model take place in real ants. This line of work, in turn, can stimulate the creation of better, more informative models.

*Ants at Work*, by Deborah Gordon, p. 142

In this chapter, I present the predictions of the present model for data from Japanese, Dutch, German, and Hindi.

#### 4.1 Japanese

In the following discussion, the verbs in nested sentences are numbered in reverse order of occurrence, i.e., the matrix verb, which appears last, is V1. The numbers do not reflect the verbs' valencies; this reverse numbering convention is merely in order to highlight the contrasting situation in Dutch (where the matrix verb is the first one seen in center embeddings).

##### 4.1.1 Gibson (1998)

Gibson (1998) reports that (25a) is less acceptable than (25b) (no information is provided in the paper as to how the results were obtained; presumably, these are intuitive acceptability judgements).

- (25) a. obasan-ga bebiisitaa-ga ani-ga imooto-o izimeta-to  
 aunt-nom babysitter-nom brother-nom sister-acc teased-comp.  
 itta-to omotteiru  
 said-comp. thinks  
 ‘The aunt thinks that the babysitter said that the elder brother teased  
 the younger sister.’
- b. bebiisitaa-ga ani-ga imooto-o izimeta-to itta-to  
 babysttr.-nom brother-nom sister-acc teased-comp. said-comp.  
 obasan-ga omotteiru  
 aunt-nom thinks  
 ‘The aunt thinks that the babysitter said that the elder brother teased  
 the younger sister.’

First, consider the application of the algorithm for (25a). For this first derivation, I incrementally build up the parse for each step. Subsequent derivations will be shown in their final form with the entire derivation history.

Step 1:

Input	Abduction/deduction	Abduction Cost	Mismatch cost
NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0

Here, given only the first NP (*obasan-ga*), a sentence with an intransitive verb (IV), denoted by  $f_1$ , is abducted. This contributes 3 to our cost so far (abduction cost, composed of the number of NPs seen so far (1), plus the number of functions abducted (1), plus the number of hypotheses abducted (1); mismatch cost is currently 0).

Step 2:

Input	Abduction/deduction	Abduction Cost	Mismatch Cost
NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
NP2-ga	$f_2(\text{NP1}, \text{NP2})$	$2+3+2=7$	0
	$f_3(\text{NP1}, f_4(\text{NP2}))$		

Given the second NP (*bebisitaa-ga*), and given that both the NPs seen so far are nominative case marked, the abducible structures are: a stative predicate taking two nominative arguments ( $f_2(\text{NP1}, \text{NP2})$ ), and a center embedded construction ( $f_3(\text{NP1}, f_4(\text{NP2}))$ ). The abduction cost here is 7: 2 NPs, 3 functions, and 2 hypotheses.

Step 3:

Input	Abduction/deduction	Abduction Cost	Mismatch Cost
NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
NP2-ga	$f_2(\text{NP1}, \text{NP2})$	$2+3+2=7$	0
	$f_3(\text{NP1}, f_4(\text{NP2}))$		
NP3-ga	$f_5(\text{NP1}, f_6(\text{NP2}, \text{NP3}))$	$3+5+2=10$	0
	$f_7(\text{NP1}, f_8(\text{NP2}, f_9(\text{NP3})))$		

We now have three nominative NPs, and so we either have an embedded stative predicate, as in  $f_5(\text{NP1}, f_6(\text{NP2}, \text{NP3}))$ , or a center embedding, as in  $f_7(\text{NP1}, f_8(\text{NP2}, f_9(\text{NP3})))$ . The abduction cost is now 10.

Step 4:

Input	Abduction/deduction	Abduction Cost	Mismatch Cost
NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
NP2-ga	$f_2(\text{NP1}, \text{NP2})$	$2+3+2=7$	0
	$f_3(\text{NP1}, f_4(\text{NP2}))$		
NP3-ga	$f_5(\text{NP1}, f_6(\text{NP2}, \text{NP3}))$	$3+5+2=10$	0
	$f_7(\text{NP1}, f_8(\text{NP2}, f_9(\text{NP3})))$		
NP4-o	$f_{10}(\text{NP1}, f_{11}(\text{NP2}, f_{12}(\text{NP3}, \text{NP4})))$	$4+3+1=8$	0

The function  $f_{10}(\text{NP1}, f_{11}(\text{NP2}, f_{12}(\text{NP3}, \text{NP4})))$  is abduced because the fourth NP is marked with accusative case, and so there must be at least one embedding with a

transitive embedded verb. The abduction cost is now 8; i.e., the model predicts that processing will take less time at this fourth NP, compared to the third NP.

Step 5:

Input	Abduction/deduction	Abduction Cost	Mismatch Cost
NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
NP2-ga	$f_2(\text{NP1}, \text{NP2})$ $f_3(\text{NP1}, f_4(\text{NP2}))$	$2+3+2=7$	0
NP3-ga	$f_5(\text{NP1}, f_6(\text{NP2}, \text{NP3}))$ $f_7(\text{NP1}, f_8(\text{NP2}, f_9(\text{NP3})))$	$3+5+2=10$	0
NP4-o	$f_{10}(\text{NP1}, f_{11}(\text{NP2}, f_{12}(\text{NP3}, \text{NP4})))$	$4+3+1=8$	0
V3	$f_{10}(\text{NP1}, f_{11}(\text{NP2}, \text{V3}(\text{N3}, \text{NP4})))$	$4+2+1=7$	2

Here, the current word, V3 is *izimeta-to*, ‘teased-complementizer’, and a deduction is performed in the following manner:

(i) V3 tries to match  $f_{10}$  in

$$f_{10}(\text{NP1}, f_{11}(\text{NP2}, f_{12}(\text{NP3}, \text{NP4}))) \Rightarrow \text{this leads to failure.}$$

This matching attempt fails because the outermost function  $f_{10}$  has a valency frame that does not match that of the actual verb.

(ii) V3 tries to match  $f_{11}$  in

$$f_{10}(\text{NP1}, f_{11}(\text{NP2}, f_{12}(\text{NP3}, \text{NP4}))) \Rightarrow \text{this also leads to failure.}$$

Here, again, the failure occurs due to the valency frame of the verb not matching that of the next function.

(iii) V3 tries to match  $f_{12}$  in

$$f_{10}(\text{NP1}, f_{11}(\text{NP2}, f_{12}(\text{NP3}, \text{NP4}))) \Rightarrow f_{10}(\text{NP1}, f_{11}(\text{NP2}, \text{V3}(\text{N3}, \text{NP4})))$$

This succeeds because the valency frame of the verb matches that of the next function. The cost now is the sum of the abduction cost (7) plus the number of failed matches (2): 9. Notice that the number of abducted functions is now 2, not 3; this is because one of the abducted functions has already been resolved by its matching with V3.

Step 6:

Input	Abduction/deduction	Abduction Cost	Mismatch Cost
NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
NP2-ga	$f_2(\text{NP1}, \text{NP2})$	$2+3+2=7$	0
	$f_3(\text{NP1}, f_4(\text{NP2}))$		
NP3-ga	$f_5(\text{NP1}, f_6(\text{NP2}, \text{NP3}))$	$3+5+2=10$	0
	$f_7(\text{NP1}, f_8(\text{NP2}, f_9(\text{NP3})))$		
NP4-o	$f_{10}(\text{NP1}, f_{11}(\text{NP2}, f_{12}(\text{NP3}, \text{NP4})))$	$4+3+1=8$	0
V3	$f_{10}(\text{NP1}, f_{11}(\text{NP2}, \text{V3}(\text{NP3}, \text{NP4})))$	$4+2+1=7$	2
V2	$f_{10}(\text{NP1}, \text{V2}(\text{NP2}, \text{V3}(\text{NP3}, \text{NP4})))$	$4+1+1=6$	1

The matching process goes as follows:

(i) V2 tries to match  $f_{10}$  in

$$f_{10}(\text{NP1}, f_{11}(\text{NP2}, \text{V3}(\text{NP3}, \text{NP4}))) \Rightarrow \text{failure.}$$

(ii) V2 tries to match  $f_{11}$  in

$$f_{10}(\text{NP1}, f_{11}(\text{NP2}, \text{V3}(\text{NP3}, \text{NP4}))) \Rightarrow f_{10}(\text{NP1}, \text{V2}(\text{NP2}, \text{V3}(\text{NP3}, \text{NP4})))$$

V2 fails to match  $f_{10}$ , but successfully matches  $f_{11}$ . The cost is now 7 (the abduction cost, 6, plus the mismatch cost, 1).

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2-ga	$f_2(\text{NP1},\text{NP2})$ $f_3(\text{NP1},f_4(\text{NP2}))$	$2+3+2=7$	0
3	NP3-ga	$f_5(\text{NP1},f_6(\text{NP2},\text{NP3}))$ $f_7(\text{NP1},f_8(\text{NP2},f_9(\text{NP3})))$	$3+5+2=10$	0
4	NP4-o	$f_{10}(\text{NP1},f_{11}(\text{NP2},f_{12}(\text{NP3},\text{NP4})))$	$4+3+1=8$	0
5	V3	$f_{10}(\text{NP1},f_{11}(\text{NP2},\text{V3}(\text{N3},\text{NP4})))$	$4+2+1=7$	2
6	V2	$f_{10}(\text{NP1},\text{V2}(\text{NP2},\text{V3}(\text{NP3},\text{NP4})))$	$4+1+1=6$	1
7	V1	$\text{V1}(\text{NP1},\text{V2}(\text{NP2},\text{V3}(\text{NP3},\text{NP4})))$	$4+0+0=4$	0

Figure 4.1: Complete derivation history for (25a); total cost = 48

Step 7 marks the end of the derivation, and is shown in Figure 4.1. The deduction in this case is immediate:

V1 tries to match  $f_{10}$  in

$$f_{10}(\text{NP1},\text{V2}(\text{NP2},\text{V3}(\text{N3},\text{NP4}))) \Rightarrow \text{V1}(\text{NP1},\text{V2}(\text{NP2},\text{V3}(\text{NP3},\text{NP4})))$$

Here, V1 matches the outermost abducted function  $f_{10}$  immediately, and the parse is completed. The cost at this stage is 4. The overall complexity of the sentence relative to other sentences is the total processing cost in the sentence. So, in this case, the maximal cost is 48. The local complexity at each point is the cost (Abduction Cost + Mismatch Cost) at that point, and constitutes a prediction about relative reading time difficulty.

By contrast, (25b)'s processing, shown in Figure 4.2, yields a lower total cost of 39. Note that in Step 5 in Figure 4.2, the appearance of an embedded verb results in an abducted hypothesis involving a matrix verb and a nominal argument. This is because V2 has the complementizer *-to*, which requires it to be an embedded verb. That is, the second clause in the definition of minimal consistency applies in this case.

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2-ga	$f_2(\text{NP1}, \text{NP2})$ $f_3(\text{NP1}, f_4(\text{NP2}))$	$2+3+2=7$	0
3	NP3-o	$f_5(\text{NP1}, f_6(\text{NP2}, \text{NP3}))$	$3+2+1=6$	0
4	V3	$f_5(\text{NP1}, \text{V3}(\text{NP2}, \text{NP3}))$	$3+1+1=5$	1
5	V2	$f_7(x, \text{V2}(\text{NP1}, \text{V3}(\text{NP2}, \text{NP3})))$	$4+1+1=6$	0
6	NP4-ga	$f_7(\text{NP4}, \text{V2}(\text{NP1}, \text{V3}(\text{NP2}, \text{NP3})))$	$4+1+1=6$	0
7	V1	$\text{V1}(\text{NP4}, \text{V2}(\text{NP1}, \text{V3}(\text{NP2}, \text{NP3})))$	$4+0+1=5$	0

Figure 4.2: Complete derivation history for (25b); total cost = 39

#### 4.1.2 Nakatani et al. (2000)

Nakatani et al. (2000) conducted several off-line acceptability rating questionnaire experiments with Japanese; their results may be summarized as follows:<sup>1</sup>

Nakatani et al. found that double embeddings are less acceptable than left branching structures. The examples below illustrate the relevant structures.

- (26) a. [obasan-wa [bebiisitaa-ga [imooto-ga naita-to] itta-to]  
 aunt-top babysitter-nom sister-nom cried-comp. said-comp.  
 omotteiru]  
 thinks  
 ‘The aunt thinks that the babysitter said that the younger sister cried.’
- b. [imooto-ga naita-to] bebiisitaa-ga itta-to] obasan-wa  
 sister-nom cried-comp. babysitter-nom said-comp. aunt-top  
 omotteiru]  
 thinks  
 ‘The aunt thinks that the babysitter said that the elder brother teased  
 the younger sister.’

---

<sup>1</sup>Note: the English glosses are sometimes different from (Nakatani et al., 2000).

The model makes the correct prediction about this set of examples, as the derivations in Figures 4.3 and 4.4 show.

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1-wa	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2-ga	$f_2(\text{NP1}, \text{NP2})$	$2+3+2=7$	0
3	NP3-ga	$f_3(\text{NP1}, f_4(\text{NP2}))$	$3+5+2=10$	0
		$f_5(\text{NP1}, f_6(\text{NP2}, \text{NP3}))$		
4	V3-to	$f_7(\text{NP1}, f_8(\text{NP2}, f_9(\text{NP3})))$	$3+2+1=6$	2
5	V2-to	$f_7(\text{NP1}, f_8(\text{NP2}, \text{V3}(\text{NP3})))$	$3+1+1=5$	1
6	V1	$\text{V1}(\text{NP1}, \text{V2}(\text{NP2}, \text{V3}(\text{NP3})))$	$3+0+1=4$	0

Figure 4.3: Complete derivation for (26a); total cost = 38

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
2	V3-to	$f_2(\text{V3}(\text{NP1}), x)$	$2+1+1=4$	0
3	NP2-ga	$f_2(\text{V3}(\text{NP1}), \text{NP2})$	$2+1+1=4$	0
4	V2-to	$f_3(\text{V2}(\text{V3}(\text{NP1}), \text{NP2}), y)$	$3+1+1=5$	0
5	NP3-ga	$f_3(\text{V2}(\text{V3}(\text{NP1}), \text{NP2}), \text{NP3})$	$3+1+1=5$	0
6	V1	$\text{V1}(\text{V2}(\text{V3}(\text{NP1}), \text{NP2}), \text{NP3})$	$3+0+1=4$	0

Figure 4.4: Complete derivation for (26b); total cost = 25

Moreover, Nakatani et al. found that in double embeddings intransitive V3's are more acceptable than transitive V3's. Examples of these structures are shown below.

- (27) a. haha-ga titi-ga fukigen-na akatyan-ga naita-to itta-to  
 mother-nom father-nom fussy baby-nom cried-comp. said-comp.  
 omotteiru  
 thinks  
 'My mother thinks that my father said that the fussy baby cried.'
- b. obasan-ga syoojiki-na bebisitaa-ga ani-ga imooto-o  
 aunt-nom honest babysitter-nom brother-nom sister-acc  
 izimeta-to itta-to omotteiru  
 teased-comp. said-comp. thinks

‘My aunt thinks that the honest babysitter said that my brother teased my sister.’

The model makes the correct prediction since (27)a has cost 40 and (27)b has cost 48; see Figures 4.5 and 4.6.

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2-ga	$f_2(\text{NP1}, \text{NP2})$ $f_3(\text{NP1}, f_4(\text{NP2}))$	$2+3+2=7$	0
3	NP3-ga	$f_5(\text{NP1}, f_6(\text{NP2}, \text{NP3}))$ $f_7(\text{NP1}, f_8(\text{NP2}, f_9(\text{NP3})))$	$3+5+2=10$	0
4	V3-to	$f_7(\text{NP1}, f_8(\text{NP2}, \text{V3}(\text{NP3})))$	$3+2+1=6$	2
5	V2-to	$f_7(\text{NP1}, \text{V2}(\text{NP2}, \text{V3}(\text{NP3})))$	$3+1+1=5$	1
6	V1	$\text{V1}(\text{NP1}, \text{V2}(\text{NP2}, \text{V3}(\text{NP3})))$	$3+0+1=4$	0

Figure 4.5: Derivation history for (27a); total cost = 38

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2-ga	$f_2(\text{NP1}, \text{NP2})$ $f_3(\text{NP1}, f_4(\text{NP2}))$	$2+3+2=7$	0
3	NP3-ga	$f_5(\text{NP1}, f_6(\text{NP2}, \text{NP3}))$ $f_7(\text{NP1}, f_8(\text{NP2}, f_9(\text{NP3})))$	$3+5+2=10$	0
4	NP4-o	$f_{10}(\text{NP1}, f_{11}(\text{NP2}, f_{12}(\text{NP3}, \text{NP4})))$	$4+3+1=8$	0
5	V3-to	$f_{10}(\text{NP1}, f_{11}(\text{NP2}, \text{V3}(\text{NP3}, \text{NP4})))$	$4+2+1=7$	2
6	V2-to	$f_{10}(\text{NP1}, \text{V2}(\text{NP2}, \text{V3}(\text{NP3}, \text{NP4})))$	$4+1+1=6$	1
7	V1	$\text{V1}(\text{NP1}, \text{V2}(\text{NP2}, \text{V3}(\text{NP3}, \text{NP4})))$	$4+0+0=4$	0

Figure 4.6: Complete derivation history for (27b); total cost = 48

#### 4.1.3 Yamashita (1997)

Yamashita (1997) investigated the effect of word order and case marking on the processing of Japanese. One of her experiments is a moving window task involving

three conditions:

A. Canonical order, with 4NPs and 2 verbs:

[NP1-nom NP2-dat [NP3-nom NP4-acc V2] V1]

B. Same structure as in Condition A, but scrambled NP3 and NP4:

[NP1-nom NP2-dat [NP4-acc NP3-nom V2] V1]

C. Same structure as in Condition A, but scrambled NP1, NP2, NP3 and NP4:

[NP2-dat NP1-nom [NP4-acc NP3-nom V2] V1]

The results for Condition A are interesting in the context of the present model;<sup>2</sup> consider the example below.

- (28) [denwa-de hansamu-na gakusei-ga sensei-ni [tumetai koibito-ga  
phone-on handsome student-nom teacher-dat cold girlfriend-nom  
nagai tegami-o yabutta-to] itta]  
long letter-acc tore-comp. said  
‘On the phone, a handsome student told the teacher that the cold-hearted  
girlfriend had torn up the letter.’

Yamashita found that reading times rose steadily in such examples until the NP preceding the accusative marked NP, and then fell at the accusative NP. The present model predicts this pattern, as shown below.

---

<sup>2</sup>In this thesis, I do not discuss the effect of word order variation since this introduces issues of pragmatics that the model currently does not take into account. The model can, however, be extended to incorporate constraints from pragmatics into the abductive process.

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1-ga	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2-ni	$f_2(\text{NP1}, \text{NP2})$	$2+1+1=4$	0
3	NP3-ga	$f_3(\text{NP1}, \text{NP2}, f_4(\text{NP3}))$ $f_5(\text{NP1}, f_6(\text{NP2}, \text{NP3}))$	$3+4+2=9$	0
4	NP4-o	$f_7(\text{NP1}, \text{NP2}, f_8(\text{NP3}, \text{NP4}))$	$4+2+1=7$	0
5	V2	$f_7(\text{NP1}, \text{NP2}, \text{V2}(\text{NP3}, \text{NP4}))$	$4+1+1=6$	1
6	V1	$\text{V1}(\text{NP1}, \text{NP2}, \text{V2}(\text{NP3}, \text{NP4}))$	$4+0+0=4$	0

Figure 4.7: Derivation for (28)

Before step 4, the reading time is predicted to rise steadily. At step 4, a fall in reading time is predicted since the number of hypotheses falls from two to one, and the number of functions is now one.

## 4.2 Dutch and German

### 4.2.1 Dutch: Kaan and Vasić (2000)

Turning next to Dutch, Kaan and Vasić (2000) conducted several self-paced reading studies and found the following. First, double embeddings are more difficult for native speakers to process than single embeddings; examples of each type are shown below:

- (29) a. De leider heeft Paul Sonya het kompas helpen leren gebruiken tijdens  
the leader has Paul Sonya the compass help teach use during  
de bergtocht  
the hike  
‘The leader helped Paul teach Sonya to use the compass during the hike.’
- b. Met aanwijzingen van de leider heeft Paul Sonya het kompas helpen  
with directions of the leader has Paul Sonya the compass teach  
gebruiken tijdens de bergtocht  
use during the hike  
‘With the leader’s directions Paul taught Sonya to use the compass during  
the hike.’

Double embeddings have a cost of 50, as shown in Figure 4.8, and the moment-by-moment reading time predictions match the observed effects.

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2	$f_2(\text{NP1},\text{NP2}), f_3(\text{NP1},f_4(\text{NP2}))$	$2+3+2=7$	0
3	NP3	$f_5(\text{NP1},\text{NP2},\text{NP3})$ $f_6(\text{NP1},f_7(\text{NP2},\text{NP3}))$ $f_8(\text{NP1},f_9(\text{NP2},f_{10}(\text{NP3})))$	$3+6+3=12$	0
4	NP4	$f_{11}(\text{NP1},f_{12}(\text{NP2},\text{NP3},\text{NP4}))$ $f_{13}(\text{NP1},f_{14}(\text{NP2},f_{15}(\text{NP3},\text{NP4})))$	$4+5+2=11$	0
5	V1	$V1(\text{NP1},f_{14}(\text{NP2},f_{15}(\text{NP3},\text{NP4})))$	$4+2+1=7$	0
6	V2	$V1(\text{NP1},V2(\text{NP2},f_{15}(\text{NP3},\text{NP4})))$	$4+1+1=6$	0
7	V3	$V1(\text{NP1},V2(\text{NP2},V3(\text{NP3},\text{NP4})))$	$4+0+0=4$	0

Figure 4.8: Derivation for (29a); total cost = 50

Single embeddings have a lower cost of 28, as Figure 4.9 shows.

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2	$f_2(\text{NP1},\text{NP2}), f_3(\text{NP1},f_4(\text{NP2}))$	$2+3+2=7$	0
3	NP3	$f_5(\text{NP1},\text{NP2},\text{NP3})$ $f_6(\text{NP1},f_7(\text{NP2},\text{NP3}))$	$3+3+3=9$	0
4	V1	$V1(\text{NP1},f_7(\text{NP2},\text{NP3}))$	$3+1+1=5$	1
5	V2	$V1(\text{NP1},V2(\text{NP2},\text{NP3}))$	$3+0+0=3$	0

Figure 4.9: Derivation for (29b); total cost = 28

Kaan and Vasić also found that RTs increased with each incoming NP, and fell at the innermost verb, which is what this model predicts. In the present model, the NP reading times are predicted to rise due to the increase in the number of abducted functions, and a fall in reading time is predicted at the first verb due to the elimination of some hypotheses.

#### 4.2.2 Dutch and German: Bach, Brown, and Marslen-Wilson (1986)

Bach et al. (1986) showed that Dutch crossed dependencies are easier to process for native Dutch speakers than German nested dependencies are for native German speakers.

Examples of crossed Dutch and nested German dependencies are shown below:

- (30) a. Jan Piet Marie zag laten zwemmen  
Jan Piet Marie saw make swim  
'Jan saw Piet make Marie swim.'
- b. ...dass Hans Peter Marie schwimmen lassen sah  
...that Hans Peter Marie swim make saw  
'...that Hans saw Peter make Marie swim.'

As discussed in Section 2.1, the Dutch SCEs are called crossed dependencies because the verbs and the subjects they link with form crossing chains (NP1 NP2 NP3 V1 V2 V3), and the German SCEs are called nested dependencies since the pattern is NP1 NP2 NP3 V3 V2 V1.

The model correctly predicts that Dutch center embeddings will be more acceptable than German ones: as shown in Figures 4.10 and 4.11, in Dutch, there will be a mismatch cost of one at the first verb seen; but in the German example, there will be a mismatch cost of five at the first verb seen.

The model incorrectly predicts that Dutch center embeddings will be as acceptable as German ones overall. However, as shown in Figures 4.10 and 4.11, in Dutch, there will be a mismatch cost of one at the first verb seen; in the German example, however, there will be a mismatch cost of five at the first verb seen. If we assume that mismatch costs are more costly (at least in German and Dutch, although this assumption causes no problems with other languages), the model predicts that reading times at the innermost verb in German will be higher than

in Dutch. Unfortunately, there exist no reading time studies on German center embeddings at present that would help us (dis)confirm this prediction.

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2	$f_2(\text{NP1},\text{NP2}), f_3(\text{NP1},f_4(\text{NP2}))$	$2+3+2=7$	0
3	NP3	$f_5(\text{NP1},\text{NP2},\text{NP3})$ $f_6(\text{NP1},f_7(\text{NP2},\text{NP3}))$ $f_8(\text{NP1},f_9(\text{NP2},f_{10}(\text{NP3})))$	$3+6+3=12$	0
4	V1	$V1(\text{NP1},f_7(\text{NP2},\text{NP3}))$ $V1(\text{NP1},f_9(\text{NP2},f_{10}(\text{NP3})))$	$3+3+2=8$	1
5	V2	$V1(\text{NP1},V2(\text{NP2},f_{10}(\text{NP3})))$	$3+1+1=5$	1
6	V3	$V1(\text{NP1},V2(\text{NP2},V3(\text{NP3})))$	$3+0+1=4$	0

Figure 4.10: Derivation for Dutch crossed embedding (30a); total cost = 41

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2	$f_2(\text{NP1},\text{NP2}), f_3(\text{NP1},f_4(\text{NP2}))$	$2+3+2=7$	0
3	NP3	$f_5(\text{NP1},\text{NP2},\text{NP3})$ $f_6(\text{NP1},f_7(\text{NP2},\text{NP3}))$ $f_8(\text{NP1},f_9(\text{NP2},f_{10}(\text{NP3})))$	$3+6+3=12$	0
4	V3	$f_8(\text{NP1},f_9(\text{NP2},V3(\text{NP3})))$	$3+2+1=6$	5
5	V2	$V1(\text{NP1},V2(\text{NP2},f_{10}(\text{NP3})))$	$3+0+1=4$	0
6	V1	$V1(\text{NP1},V2(\text{NP2},V3(\text{NP3})))$	$3+0+1=4$	0

Figure 4.11: Derivation for German nested embedding (30b); total cost = 41

### 4.3 Hindi: Vasishth (to appear, 2002)

In (Vasishth, to appear, 2002) I describe a self-paced reading time study which show the following. First, in center embeddings, accusative case marking on direct objects in Hindi (which marks definiteness in the case of inanimate objects), results

in increased processing difficulty. Examples of single center embeddings are shown below.

- (31) a. Siitaa-ne Hari-ko [kitaab khariid-neko] kahaa  
 Sita-erg Hari-dat book buy-inf told  
 ‘Sita told Hari to buy a/the book.’
- b. Siitaa-ne Hari-ko [kitaab-ko khariid-neko] kahaa  
 Sita-erg Hari-dat book-acc buy-inf told  
 ‘Sita told Hari to buy the book.’

The model predicts that in the case of both (31a,b) there will be only one hypothesis by the time the third NP is processed, However, as in the case of Joshi’s EPDA and Lewis’ ICT, the effects of discourse context can be incorporated into the model since no assumptions are made in the current model about pragmatic information.

Step	Input	Abduction/deduction	Abduction Cost	Mismatch Cost
1	NP1-ne	$f_1(\text{NP1})$	$1+1+1=3$	0
2	NP2-ko	$f_2(\text{NP1},\text{NP2})$	$2+1+1=4$	0
3	NP3(-ko)	$f_3(\text{NP1},f_4(\text{NP2},\text{NP3}))$	$3+2+1=6$	0
4	V2	$f_2(\text{NP1},V2(\text{NP2},\text{NP3}))$	$3+1+1=5$	1
5	V1	$V1(\text{NP1},V2(\text{NP2},\text{NP3}))$	$3+0+0=3$	0

Figure 4.12: Derivation for Hindi examples (31a,b)

Second, the Hindi adverb-insertion facts are also consistent with the model. If an adverb is present after the final NP, the likelihood of a verb appearing next (as opposed to another NP) increases, thereby resulting in faster reading time at the verb.

Finally, the model predicts that a ditransitive verb following four NPs will be read faster than a transitive verb. This is because the ditransitive verb hypothesis is

ordered before the transitive verb hypothesis, and should therefore result in a lower mismatch cost.

- (32) a. Siitaa-ne Hari-ko Ravi-ko kitaab de-neko kahaa  
Sita-erg Hari-dat Ravi-dat book give-inf told  
'Sita told Hari to give a book to Ravi.'
- b. Siitaa-ne Hari-ko Ravi-ko kitaab paḍh-neko bol-neko kahaa  
Sita-erg Hari-dat Ravi-dat book read-inf tell-inf told  
'Sita told Hari to tell Ravi to read a book.'

This predicted difference was verified in a between-subjects reading time study (Vasishth, to appear, 2002).

## CHAPTER 5

### A PROLOG IMPLEMENTATION OF THE MODEL

This chapter describes an implementation of the abductive-inference driven parser written in Prolog. The goal of this chapter is to assist the reader in navigating the Prolog code in Appendix 2. The current version of the entire code is downloadable from the website <http://www.ling.ohio-state.edu/~vasishth/AIM/>. The code should run without modification on any “standard” Prolog system; it has been written using the freely available SWI-Prolog implementation (<http://www.swi-prolog.org/>). The discussion below presupposes at least a reading knowledge of Prolog. This can be easily acquired by consulting any of a number of excellent textbooks, e.g., (Clocksin & Melish, 1981). The reader unfamiliar with Prolog but familiar with programming in general should at least read the brief tutorial at <http://cbl.leeds.ac.uk/~tamsin/prologtutorial/> before proceeding further.

#### 5.1 The design of the parser: Some caveats

The parser presented here is rather unorthodox, given that the underlying theory is so similar to the standard left-corner parser. The reader familiar with parsers written in Prolog may wonder why Definite-Clause Grammar (DCG) is not used, as is standardly done (Saint-Dizier, 1994).

This parser, however, is designed to closely follow every step of the algorithms described earlier. A central goal was to maintain a clear connection between the sentence parsing theory and the parser itself. Although it is certainly possible to transform the present parser into a DCG-type parser, such a transformation completely obscures the underlying theory.

## 5.2 The main call to the program

In order to parse a Japanese sentence such as (33)

- (33) taro-ga jiro-ga sinda-to omotteiru  
taro-nom jiro-nom died-comp thinks  
'Taro thinks that Jiro died.'

the Prolog clause in (34) is invoked. The bracketings within the sentence demarcate the self-paced reading presentation units.

- (34) parse([[taro,ga],[jiro,ga],sinda-to,omotteiru],japanese).

The top-level call is to the rule in Figure 5.1:

```
parse(S,Lang):- load_hypothesis_formers_for(Lang),
                assertz(input_so_far([])),
                assertz(working_memory([])),
                write('Initializing working memory...'),
                nl,nl,
                write('Starting processing...'),
                nl,nl,
                write('The input sentence is: '),
                write(S),
                nl,
                process(S,Lang).
```

Figure 5.1: The main call to the parser

First the rule initializes `working_memory` and `input_so_far` (both are dynamically changing list data structures). Then it processes the sentence incrementally, phrase by phrase.

### 5.3 The `process/2` rule

The `process/2` rule incrementally consumes the input sentence, and succeeds after the input string is empty (the base case in Figure 5.2). By “incrementally consumes”, I mean that if the sentence has the structure NP1 NP2 NP3 V2 V1, first NP1 is processed, then NP1 NP2, and so on. This simulates the incremental presentation of the material.

The first step in `process/2` is to determine whether the word is a noun phrase (NP) or a verb; based on this, either the abductive step is carried out (if it is an NP), or an attempt to match begins (if it is a verb). These are shown as Case 1 and Case 2 in Figure 5.2.

The important predicates here are `matchverb` and `abduce_from`; the rest are either book-keeping predicates or rules, the functions of which should be clear from their names and the documented code in Appendix 2. We look at `matchverb` and `abduce_from` in the next two sections.

### 5.4 The rule `matchverb`

Figure 5.3 shows the rule called by `process`. Assume that some hypotheses have been abduced already (how this is done is discussed in the next section). If the first hypothesis in working memory happens to be an embedded clause and a matrix verb is seen next by the parser, the verb’s lexical information is unified with the hypothesis

```

%%%process(+Sentence,+Language)
%%%=====
%%%
%%% Case 1: next item is a verb; the verb must match an abduced
%%%         function
%%% Case 2: next item is an NP; try to match it with abduced
%%%         hypotheses' arguments

%%% Base case:

process([],_):- true.

%%% Case 1:

process([First|Rest],Lang):-
    is_verb(First),
    working_memory(WM),
    matchverb(First,WM),
    write_match_succeeded(First),
    process(Rest,Lang).

%%% Case 2:

process([First|Rest],Lang):-
    is_np(First),
    update_input_so_far([First],NewInputSoFar),
    retract(working_memory(_)),
    abduce_from(NewInputSoFar),!,
    process(Rest,Lang).

```

Figure 5.2: The call to process/2

and working memory is updated to reflect the fact that the verb has been instantiated to a predicted verb in a hypothesis.

```
matchverb(Verb, [hyp(Verb,matrixverb,Args)|TailWM]):-
    lexicon(Verb,verb,matrixverb,_),
    retract(working_memory(_)),
    append([hyp(Verb,matrixverb,Args)],TailWM,NewWM),
    assertz(working_memory(NewWM)).
```

Figure 5.3: Processing in the case where a matrix verb is seen

Figure 5.4 shows the case where the current verb being processed is not a matrix verb, and the predicted matrix verb in the hypothesis being considered currently does *not* match the verb. The current verb could be an embedded verb, if the verb's arguments (available from its lexical entry) match with the NPs seen so far. A new rule, `matchverb3`, is used to search hypotheses embedded inside another hypothesis (a new rule is used here only to keep the program in a state that is easier to maintain—one could make the `matchverb` more complex so that it would deal with all situations involving matching). The rule `find_hyp/3` simply returns (in its last argument, “Hyp” in Figure 5.4) a list of hypotheses embedded inside a main hypothesis; these embedded hypotheses need to be searched recursively using `matchverb3`.

If `matchverb3` fails, the parser moves on to the next available clause for the main call to `matchverb`. If `matchverb3` succeeds, the current verb needs to be unified with the predicted verb in an embedded hypothesis; this is done explicitly using the rule `my_unify`. After that, working memory is updated to reflect its new state.

Figure 5.5 presents the next set of rules. It could happen that the current verb simply matched a hypothesis fully; in that case we just update working memory to reflect this fact. If the first hypothesis in working memory fails, a recursive

```

matchverb(Verb, [hyp(Verb1, matrixverb, Args) | TailWM]) :-
    find_hyp(Args, [], Hyp),
    !,
    matchverb3(Verb, Hyp, Args),
    my_unify(Hyp, Args),
    retract(working_memory(_)),
    append([hyp(Verb1, matrixverb, Args)], TailWM, NewWM),
    assertz(working_memory(NewWM)).

```

Figure 5.4: Processing a non-matrix verb

call examines the rest of working memory. Finally, if every hypothesis in WM has been examined and nothing matches, `matchverb` signals that the sentence is not grammatical and writes an exception. This exception-rule will be invoked with ungrammatical input like

```

(35) taro-ga   jiro-ga   sinda
      Taro-nom Jiro-nom died
      ...

```

```

matchverb(Verb, [hyp(Verb, Type, Args) | _TailWM]) :-
    lexicon(Verb, verb, Type, Args),
    retract(working_memory(_)),
    assertz(working_memory([hyp(Verb, Type, Args)])) .

%Recursive call:
matchverb(Verb, [_Failedhyp | TailWM]) :- matchverb(Verb, TailWM) .

%Exception:
matchverb(_Verb, []) :- write_exception.

```

Figure 5.5: The `matchverb` rule: The simple case

Next, we turn to the rule `matchverb3`, which is called by `matchverb`.

### 5.4.1 The rule `matchverb3`

As shown in Figure 5.6, the base case of this recursive rule is that if there are no more embedded hypotheses, it failed to match the main hypothesis, so it should succeed vacuously (it may sound counter-intuitive that the clause succeeds when it fails, but this clause needs to succeed in order to allow the recursion in the main call to `matchverb` to continue; it is the main call that decides whether there was a total failure to match any hypothesis).

The main call is the second rule in Figure 5.6; if that fails, the third, recursive call is invoked in order to examine the remaining embedded hypotheses.

```
matchverb3(_Verb, [], _):- true.

matchverb3(Verb, [hyp(Verb, Type, Args) | _TailWM], _):-
    lexicon(Verb, verb, Type, Args).

%Recursive call:
matchverb3(Verb, [_FailedHyp | TailWM], _):-
    matchverb3(Verb, TailWM, _).
```

Figure 5.6: The rule `matchverb3`

## 5.5 The rule `abduce_from`

As shown in Figure 5.7, this rule looks up the list of hypothesis formers for the language specified at the beginning of the parse, and then calls `abduced_hyps/4`, which recursively examines the list of hypothesis formers and updates working memory with the abduced hypotheses.

As shown in Figure 5.8, the predicate `abduced_hyps/4` recursively examines the list of hypothesis formers for a match, and updates working memory in its base case

```

abduce_from(InputList):-
    hypothesis_formers(_,H),!,
    abduced_hyps(InputList,H,[],_).

```

Figure 5.7: The rule `abduce_from`

(when the list of hypothesis formers has been searched exhaustively). This predicate calls another predicate, `get_args/3`, which is described with reference to Figure 5.9.

```

% Base case:
abduced_hyps(InputList,[],WM,WM):-assertz(working_memory(WM)),
    write_to_log(InputList,WM),
    nl.

abduced_hyps(InputList,[hyp(Instance,Vtype,ArgsList)|T],TempWM,_):-
    get_args(InputList,hyp(Instance,Vtype,ArgsList),ListofHyps),
    append(TempWM,ListofHyps,NewWM),
    abduced_hyps(InputList,T,NewWM,_).

abduced_hyps(InputList,[_Failed|T],TempWM,_):-
    abduced_hyps(InputList,T,TempWM,_).

```

Figure 5.8: The rule `abduced_hyps`

As shown in Figure 5.9, in the simple case, all the items in the input list match the arguments in the current hypothesis being considered. The second clause in Figure 5.9 simply ends the search for arguments if the input list consists of one or less (zero) items. This is necessary because we are recursively searching for argument matches.

More complex is the situation where the arguments in the hypothesis cannot saturate the input list of NPs and the hypothesis is not a monoclausal structure. In this case, the parser does an incremental search for argument matches. This essentially deals with a situation where one of the input items, say the first NP, fits a slot in a hypothesis, but the rest of the items (e.g., other NPs) fit with slots in

embedded clauses. Such a search for matches requires an incremental search through the hypothesis' arguments.

```
% Simple case:
get_args(InputList,hyp(Instance,V,ArgsList),
         [hyp(Instance,V,ArgsList)]):-
         InputList = ArgsList.

get_args(InputList,_,EmptyList):- length(InputList,X),
                                   X =< 1,
                                   EmptyList = [].

%Complex case:
get_args(InputList,hyp(Instance,matrixverb,ArgsList),
         ListofHyps):-
         incrementalmatch(InputList,
         hyp(Instance,matrixverb,ArgsList),ListofHyps).
```

Figure 5.9: The rule `get_args`

Figure 5.10 shows the rules for incremental search. As discussed above, `incremental_match/3` simply tries to match the input items with the arguments of a hypothesis, proceeding incrementally in the sense that if it finds a matching NP-slot, it proceeds to look further in the hypothesis for matching slots for the remaining items in the input list.

## 5.6 Some sample runs

In this section, I show some sample runs of the parser, using Japanese and Dutch as examples. The German and Hindi structures will work like Japanese, since they involve nested structures.

First we consider Japanese. For a simple, monoclausal sentence like (36)

```

incrementalmatch([],_,_).

incrementalmatch([H|T],hyp(Instance,Vtype,[H1|TArgs]),
                 [hyp(Instance,Vtype,[H1|TArgs])]):-
    H = H1,
    hypothesis_formers(_,HF),
    search_hf(T,HF,TArgs).

search_hf(_T,[],_).

search_hf(RestInputList,[hyp(Instance,VerbType,Args)|_Tail],Result):-
    RestInputList = Args,
    [[hyp(Instance,VerbType,Args)]] = Result.

search_hf(T,[_Failed|Tail],_Hypothesis):-
    search_hf(T,Tail,_Hypothesis).

```

Figure 5.10: The rules for incremental search

```

(36) taro-ga jiro-ga sukida
      taro-nom jiro-nom likes
      'Taro likes Jiro.'

```

the call to the parser and the resulting output will be as in Figure 5.11 (the symbol “?-” is the Prolog command prompt):

```

?- parse([[taro,ga],[jiro,ga],sukida],japanese).

Initializing working memory...

Starting processing...

The input sentence is: [[taro, ga], [jiro, ga], sukida]

For input: [[taro, ga]]

  Hypotheses generated are: [hyp(_G350, intrverb, [[taro, ga]])]

For input: [[taro, ga], [jiro, ga]]

  Hypotheses generated are:

[hyp(_G553, stativeverb,
      [[taro, ga], [jiro, ga]]),
 hyp(_G578, matrixverb,
      [[taro, ga],
       [hyp(_G629, intrverb, [[jiro, ga]])]])]

The verb sukida matched a hypothesis.
Current state of working memory memory is:
[hyp(sukida, stativeverb, [[taro, ga], [jiro, ga]])]

```

Figure 5.11: Trace showing working memory changes for (36)

For a single embedding like (37)

- (37) taro-ga jiro-ga sinda-to omotteiru  
taro-nom jiro-nom died-comp thinks  
'Taro thinks that Jiro died.'

the call to the parser and the resulting output will be as shown in Figure 5.12.

Starting processing...

The input sentence is: [[taro, ga], [jiro, ga], sinda-to,  
omotteiru]

For input: [[taro, ga]]

Hypotheses generated are: [hyp(\_G356, intrverb, [[taro, ga]])]

For input: [[taro, ga], [jiro, ga]]

Hypotheses generated are: [hyp(\_G559, stativeverb, [[taro, ga],  
[jiro, ga]]),  
hyp(\_G584, matrixverb, [[taro, ga],  
[hyp(\_G635, intrverb, [[jiro, ga]])]])]

The verb sinda-to matched a hypothesis.

Current state of working memory memory is:

[hyp(\_G915, matrixverb, [[taro, ga], [hyp(sinda-to, intrverb,  
[[jiro, ga]])]])]

The verb omotteiru matched a hypothesis.

Current state of working memory memory is:

[hyp(omotteiru, matrixverb, [[taro, ga], [hyp(sinda-to,  
intrverb, [[jiro, ga]])]])]

Figure 5.12: Trace showing working memory changes for (37)

Turning next to Dutch, for a single embedding like (38),

- (38) Jan Piet zag zwemmen  
Jan Piet saw swim  
'Jan saw Peter swim.'

the call to the parser and the resulting output will be as shown in Figure 5.13.

```
?- parse([[jan,nom],[piet,nom],zag,zwemmen],dutch).  
  
The input sentence is: [[jan, nom], [piet, nom], zag, zwemmen]  
  
For input: [[jan, nom]]  
  
Hypotheses generated are: [hyp(_G303, intrverb, [[jan, nom]])]  
  
For input: [[jan, nom], [piet, nom]]  
  
Hypotheses generated are: [hyp(_G417, matrixverb, [[jan, nom],  
[hyp(_G457, intrverb, [[piet, nom]])]])]  
  
The verb zag matched a hypothesis.  
Current state of working memory memory is:  
[hyp(zag, matrixverb, [[jan, nom], [hyp(_G640, intrverb,  
[[piet, nom]])]])]  
  
The verb zwemmen matched a hypothesis.  
Current state of working memory memory is:  
[hyp(zag, matrixverb, [[jan, nom], [hyp(zwemmen, intrverb,  
[[piet, nom]])]])]
```

Figure 5.13: Trace showing working memory changes for (38)

## CHAPTER 6

### CONCLUDING REMARKS

All models are wrong. Some models are useful.

(Attributed to George Box)

This thesis presented a hybrid abductive/deductive model of human language processing, based on existing psycholinguistic results. As discussed in the preceding chapter, it makes more accurate predictions about processing difficulties than several existing models. An important issue is that many of the mechanisms proposed have correlates in these other theories. For example, the number of NPs seen up to a given point are counted as part of the abduction cost; this corresponds to the number of discourse referents, which is a critical component of Gibson's model (although Gibson's model only considers *new* discourse referents). These kinds of correspondence naturally raise the question, what is new about the present parser? The main contribution of the present model is that it employs a very general perceptual mechanism—abduction—to constrain incremental parsing decisions.

Empirically, the model fares better than existing accounts for the four languages under consideration. For example, none of the existing theories can currently account for the fall in reading times at the accusative verb in Japanese and at the first verb in Dutch; and Gibson's model (Gibson, 2000) appears to make incorrect predictions for the increased reading times for verbs (see (Kaan & Vasić, 2000) for details). However, it remains to be seen whether all the predictions the model makes are borne out.

For example, the model predicts that there will be a fall in reading time when the number of abducted hypotheses is reduced to one in working memory as a result of new incoming information. This happens to be the correct prediction for Yamashita’s Japanese data, Kaan and Vasić’s Dutch data, and the Hindi facts, but we do not have enough data yet to determine whether this prediction is borne out (for example) for (25b), repeated below as (39).

- (39) bebiisitaa-ga ani-ga imooto-o izimeta-to itta-to obasan-ga  
 babysitr.-nom brother-nom sister-acc teased-comp. said-comp. aunt-nom  
 omotteiru  
 thinks  
 ‘The aunt thinks that the babysitter said that the elder brother teased the  
 younger sister.’

Further, I currently do not have a precise account for the scrambling facts (e.g., those presented in (Yamashita, 1997)). One reason for hesitating to extend this model so as to account for scrambling is that word order variation is almost always correlated with a particular discourse context. In spite of this, studies on scrambling and processing like (Yamashita, 1997) and (Vasishth, to appear, 2002) focus on the processing of a scrambled sentence presented to subjects without any discourse context. It is unclear whether such no-context scrambled sentences can be fairly compared with their unscrambled correlates. Note that none of the models are currently able to account for the scrambling facts in Hindi; the Hindi results suggest that Gibson’s integration distance and the EIC (although probably an important factor in general) may be overridden by other discourse factors, and existing models haven’t systematically taken this possibility into account. We must therefore await further empirical work before any valid conclusions can be drawn about the processing of scrambled sentences.

Finally, ways for incorporating the proposed model (which I will call AIM, the Abductive Inference Model) into existing models are briefly considered in the following sections. Incorporating the model into existing ones is a desirable step since each of these models have other attractive properties, as discussed below.

## 6.1 The EPDA and AIM

The connection between the current model and the EPDA is quite straightforward: Both are pushdown automata. What is different in Joshi's model is that NPs are stored as-is in the stack(s). If we assume that the units stored in the stack are hypotheses, not simple NPs, and adopt the complexity metric presented here, Joshi's model can still be used as a model for the processing facts. One might object that the processing explanations fall out of the abductive-inference driven parsing strategy along with the complexity metric, so the EPDA does not appear to contribute anything. However, the EPDA provides the storage mechanism (which happens to correspond to the mechanism used in AIM). Furthermore, even in its original form the EPDA relies on a complexity metric that is not a necessary consequence of the computational architecture; thus, one could question the contribution of the EPDA *per se* in explaining the processing facts.

## 6.2 The EIC and AIM

As discussed earlier, a head-driven parsing strategy falls out of the Mother Node Construction constraint and the Axiom (Section 2.2.3). Instead, if one treats node-construction as a predictive step that the parser makes, and node-recognition as a confirmatory step, we have AIM's parsing strategy.

However, there are some crucial differences between EIC and AIM. In EIC, no cost is associated with the node-construction step *per se*; rather, the node-recognition step determines the IC-to-word ratio. In AIM, by contrast, abductive inferences result in differing numbers of hypotheses being stored in working memory, a larger number of hypotheses increasing the load on memory resources. Moreover, in EIC the node-construction is highly category-specific. For example, a noun will construct an NP phrasal category, not an S above it as well. By contrast, AIM's parsing mechanism directly constructs full sentence-level hypotheses from items like nouns and NPs.

It is straightforward to incorporate the parsing model into EIC by altering the parsing strategy so that it is (a variant of) arc-eager LR parsing, instead of head-driven parsing. Less straightforward is the strong claim that EIC is the only determiner of complexity, since it is clear (from the Hindi experiments) that new- and given-ness affect processing.

### 6.3 The DLT and AIM

The present model is very similar to Gibson's DLT, but there are important differences, which have empirical consequences. In the DLT, the complexity metric is defined over the number of new discourse referents, discourse factors such as given/new information, and the distance between heads and dependents. However, the DLT incorporates constraints independent of the parsing strategy. Given the empirical problems for the DLT discussed in the Chapter 2 and in (Vasishth, to appear, 2002), these differences suggest obvious modifications that could be made to Gibson's model so as to improve its cross-linguistic empirical coverage. Quite simply, the present parser can be incorporated without modification into the DLT, and the

resulting model would have all the characteristics of DLT as well as AIM.

## 6.4 ICT and AIM

The connection between the present model and the ICT is best understood with reference to a newer ACT-R version of the ICT which I became aware of very recently. I will call this new version ICT-02, to distinguish it from the version described in (Lewis & Nakayama, 2001). The discussion below is based entirely on conversations (January 2002) and collaborative research with Richard Lewis regarding ICT-02.

As items (e.g., nouns, verbs) are processed in real time, they are stored in working memory by being associated with some features (for nouns, features like, “is nominal”, “has accusative case”, etc.). When an item is encountered, it can make a RETRIEVAL REQUEST: the item examines the entities currently in working memory and sets certain RETRIEVAL CUES. E.g., a transitive verb would set a retrieval cue for an entity having the features of a transitive verb. When a retrieval request is made by setting retrieval cues, interference results between the different candidate entities active in working memory. The amount of interference depends on the degree of similarity of the candidates to the retrieval requests.

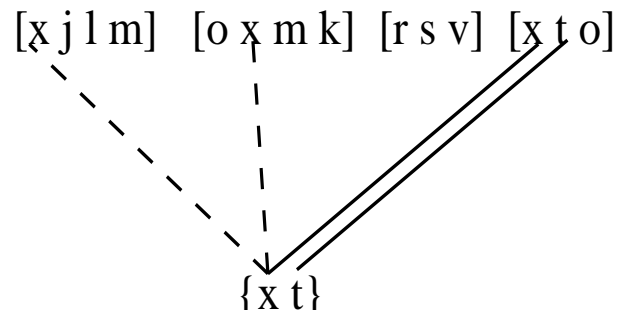


Figure 6.1: Schematic illustration of interference

Interference can be illustrated schematically as in Figure 6.1. Here, the item in parentheses ( $\{x\ t\}$ ) denotes a retrieval cue consisting of syntactic features “x” and “t”. For example, this could be the retrieval cue set by a nominative noun phrase: “x” could be a feature like “this noun phrase is a subject of a sentence”, and “t” might specify the features of the sentence node. The items in square brackets are feature-based encodings of actual lexical items, e.g., verbs that might be encountered during the course of a parse. The current word makes a retrieval request and the candidate items in working memory become retrieval cues that compete with each other for matching with the retrieval request. The best match is the item with the greatest number of common features (here,  $[x\ t\ o]$ ), and in case of a tie between two items, the better match is determined by the relative activation levels of the two retrieval cues, the one with the higher activation level having a better match.

There are two kinds of retrieval requests. One involves an attempted match with a predicted structure, and the other is an attempted match with an actual structure. An example of the former is the case where a nominative NP has been seen; here, a sentence-level node S is predicted. If a verb appears next, the lexical entry for the verb makes a retrieval request, i.e., it looks for an S in working memory, and attempts to match its own S-features with that of the predicted S. An example of the latter kind of retrieval request is the case where a nominative NP has been seen. The NP sets the retrieval cue as above, but if a PP appears next instead of a verb, the PP makes a retrieval request for a noun.

An important difference between the earlier version of ICT and ICT-02 is that the parser incorporates a predictive component. This has the following consequence. As discussed in Section 2.4.2, in sentences like (40) at the innermost verb, the final NP was assumed to be immune to the effects of proactive interference since it was

out of the focus of attention. Recall that focus of attention is assumed to include the current item and the immediately preceding one, so at the innermost verb the final NP is no longer in the focus of attention.

- (40) Ani-ga                    sensei-ni    onna-no-ko-ga asa        rokuji kara  
elder brother-nom teacher-dat girl-nom        morning 6AM from  
asondeiru-to renrakusita  
playing-that notified  
'My older brother notified the teacher that a girl has been playing since 6AM.'

By contrast, in ICT-02, a verb is predicted by the NPs seen so far and each prediction has an activation level associated with it (it is integral to the ACT-R architecture that items being processed have a certain activation). When the adverb is seen, it is integrated with this prediction, and this has the consequence that the base-level activation of the predicted verb increases, and therefore when the verb is seen, the retrieval of the prediction is easier (compared to the case where no adverb was present) due to its higher activation level. This is because the retrieval of the prediction is easier due to its higher base-level activation. This is precisely the result found for Hindi (Vasishth, to appear, 2002).

The connection with AIM is straightforward: instead of predictions for verbs, we can have abducted hypotheses. The parsing strategy of AIM can be used as-is in ICT-02, while retaining all the other characteristics of ICT-2 (e.g., interference effects) that are clearly well-motivated empirically.

## APPENDIX 1: LEFT-CORNER PARSING AND COGNITIVE MODELING

This appendix outlines the basis for preferring left-corner (or LR) parsing as the appropriate model for human parsing. I begin by summarizing some very basic facts about parsing in general, and then present the motivation for left-corner parsing as a strongly equivalent model of human parsing. The parsing theory discussed is far from a comprehensive review of all of parsing theory; for such an overview, see (Aho & Ullman, 1972), (Aho & Ullman, 1973). Rather, the goal here is to demonstrate that the parsing model presented in this thesis is well-motivated, and has the potential for providing a broad-based explanation for cross-linguistic facts about human sentence processing.

A parsing algorithm for a grammar  $\mathcal{G}$  takes an input string  $s$  and produces as output either (a) a parse tree for  $s$  if  $s \in \mathcal{L}(\mathcal{G})$ , i.e., if  $s$  is a sentence of the language generated by  $\mathcal{G}$ ; or (b) returns an error message if  $s \notin \mathcal{L}(\mathcal{G})$ . There are three broad types of parsing algorithms for context-free grammars (Hopcroft & Ullman, 1979): TOP-DOWN, where parse trees are built from the root to the terminals; BOTTOM-UP, where parse trees are built from the terminals to the root; and HYBRID (bottom-up and top-down). Simplified versions of these three strategies are illustrated in the figure below (this presentation is adapted from (Abney & Johnson, 1991) and (Resnik, 1992)). The terminals are C, D, and E, the root is A, and there is one nonterminal, B.

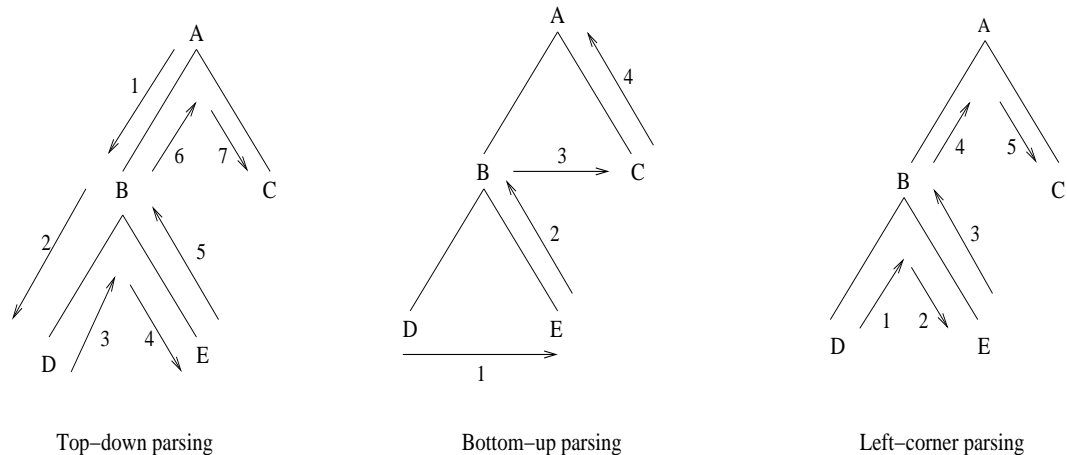


Figure A1.1: The three main types of parsing strategy

In top-down parsing, the tree is built from the root to nonterminal nodes, the terminals being built after a nonterminal mother node is built. This is a predictive parsing method, since the root and nonterminals in effect predict the existence of the terminals. By contrast, bottom-up parsing proceeds by first building all the terminal nodes in a (sub)tree before the root of the (sub)tree is built. Thus the pure bottom-up parser involves no prediction of nonterminal nodes. The third type is hybrid parsing, where the leftmost child of a nonterminal is built before the nonterminal (the bottom-up part), and the siblings of the leftmost child are built after the nonterminal (the top-down part). This is called the left-corner parser, and incorporates both non-predictive (or data-driven) and predictive components.

Other variations are possible for all these types of parser. However, the important question is: is one or the other parsing method a realistic model of human parsing? As discussed in detail by Johnson-Laird (1983, 296-309), (Abney & Johnson, 1991), and (Resnik, 1992), purely top-down or purely bottom-up strategies turn out to be inappropriate models for human parsing since they are unable to capture the

Strategy	Left branching	Center embedding	Right branching
Top-down	$O(n)$	$O(n)$	$O(1)$
Bottom-up	$O(1)$	$O(n)$	$O(n)$
Left-corner	$O(1)$	$O(n)$	$O(1)$

Figure A1.2: Space complexity for the constructions in (41)

observation by Miller and Chomsky (1963, 468-470) that left-branching and right-branching structures are relatively easy to process compared to center embeddings:

- (41) a. Bill's book's cover is dirty. (left branching)  
b. Bill has the book that has the cover that is dirty. (right branching)  
c. The rat the cat the dog chased killed ate the malt. (center embedding)

Figure A1.2 summarizes the space complexity associated with each type of construction (as a function of the length  $n$  of the input string) Resnik (1992). Partly due to these facts, many researchers have proposed that some variant or the other of left-corner parsing is the appropriate model for human parsing (Marcus (1980), Shieber (1983), Pereira (1985), Resnik (1992), Covington (1994), Schlesewsky et al. (no date), Steedman (2000)).

## APPENDIX 2: PROLOG CODE

```
/*=====
                                     A Parser Based on Abductive Inference
                                     Shravan Vasishth
=====*/

/*-----
                                     ANCILLIARY FILES
The files contain prolog rules needed by the main program.
The code in these files is presented in this appendix under
appropriate headers.
-----*/

:-[test_sentences].

:-[abduction_predicates_recursive].

:-[matchverbpredicates].

:-[myunify].

/*-----
                                     TEST SENTENCES
-----*/

%%%Japanese
%%%=====

japanese1 :- parse([[taro,ga],[jiro,ga],sinda-to,omotteiru],japanese).

%%%Dutch
%%%=====

dutch1 :- parse([[jan,nom],[piet,nom],zag,zwemmen],dutch).
```

```
/*-----  
DYNAMICALLY CHANGING PREDICATES  
----- */
```

```
%%%working_memory/1  
%%%=====  
%%%  
%%% A list containing hypotheses of the following type  
%%% (Japanese example): hyp(intrverb,[[_NP1,ga]]).  
%%% This is a hypothesis predicting an intransitive verb that  
%%% takes one nominative (-ga) case-marked argument.
```

```
:-dynamic(working_memory/1).
```

```
%%%input_so_far/1  
%%%=====  
%%%  
%%% A list containing the phrases of a sentence so far.
```

```
:-dynamic(input_so_far/1).
```

```
%%%hypothesis_formers/2  
%%%=====  
%%%  
%%% This is language-specific. The first argument specifies the  
%%% language (japanese, hindi, german, dutch), and the second  
%%% argument is a list containing hypotheses like  
%%% hyp(_VerbInstance, intrverb,[[_NP1,ga]]) (for Japanese).
```

```
:-dynamic(hypothesis_formers/2). % language specific
```

```

/*=====
                                MAIN PROCESSING LOOP
=====*/

%%%parse(+Sentence,+Language)
%%%=====
%%%
%%% Load hypothesis formers for the language specified,
%%% initialize input_so_far to empty list,
%%% initialize working memory to empty list,
%%% start processing sentence.
%%% Example of a call: parse([[taro,ga],[jiro,ga],sukida],
%%%                               japanese).

parse(S,Lang):- load_hypothesis_formers_for(Lang),
                assertz(input_so_far([])),
                assertz(working_memory([])),
                write('Initializing working memory...'),
                nl,nl,
                write('Starting processing...'),nl,nl,
                write('The input sentence is: '),
                write(S),
                nl,
                process(S,Lang).

%%%load_hypothesis_formers_for(+Language)
%%%=====
%%%
%%% Loads hypotheis formers for the specified language.

load_hypothesis_formers_for(japanese):-
    assertz(hypothesis_formers(japanese,
        [hyp(_, intrverb,[[_NP1,ga]]),
        hyp(_, trverb,[[_NP1, ga],[_NP2,wo]]),
        hyp(_, stativeverb,[[_NP1, ga],[_NP2,ga]]),
        hyp(_, matrixverb,[[_NP1, ga],
            [_EmbeddedClause]])])).

```

```

%%%process(+Sentence,+Language)
%%%=====
%%%
%%% Case 1: next item is a verb; the verb must match an
%%%         abduced function
%%% Case 2: next item is an NP; try to match it with
%%%         abduced hypotheses' arguments

process([],_):- true,!.

%%% Case 1:

process([First|Rest],Lang):- is_verb(First),
                             working_memory(WM),
                             matchverb(First,WM),
                             write_match_succeeded(First),
                             process(Rest,Lang).

%%% Case 2:

process([First|Rest],Lang):- is_np(First),
                             update_input_so_far([First],
                                                  NewInputSoFar),
                             retract(working_memory(_)),
                             abduce_from(NewInputSoFar),!,
                             process(Rest,Lang).

```

```

/*=====
                                INPUT-OUTPUT AND OTHER RULES
=====*/

%%%is_verb(+List)
%%%=====
%%%
%%% Checks if head of List is a verb.

is_verb(Verb):- lexicon(Verb,verb,_Type,_ArgsList),!.

%%%is_np(+List)
%%%=====
%%%
%%% Checks if head of List is an NP.

is_np([H|_T]):- lexicon(pn,H).

is_np([H|_T]):- lexicon(n,H).

%%%get_hyp_formers(+Language,-HypothesisFormers)
%%%=====
%%%
%%% Returns the list of hypothesis formers for the
%%% language specified.

get_hyp_formers(Lang,H):-hypothesis_formers(Lang,H).

%%%update_input_so_far([+CurrentWord],-UpdatedInputSoFar)
%%%=====
%%%
%%% Returns a list containing the total input so far.

update_input_so_far([First],NewInputSoFar):-
    input_so_far(InputSoFar),
    append(InputSoFar,[First],
           NewInputSoFar),
    retract(input_so_far(InputSoFar)),
    assertz(input_so_far(NewInputSoFar)).

```

```

%%%append(+List1,+List2,-List1List2)
%%%=====
%%%
%%% Given two lists, returns an appended list.

append([],X,X).
append([X|Y],Z,[X|R]) :- append(Y,Z,R).

%%%writelanguage/2
%%%=====
%%%
%%%Writes language to standard output; called by
%%%write_exception/0 (below).

writelanguage(japanese,'Japanese').

%%%write_match_succeeded/1
%%%=====
%%%
%%%For each verb, if a match is found in working memory,
%%%prints the updated
%%%working memory state.

write_match_succeeded(Verb):-
    nl, write('The verb '),
    write(Verb),
    write(' matched a hypothesis. '),nl,
    write('Current state of working memory memory is: '),
    nl,
    working_memory(W),
    write(W),
    nl.

write_exception:- write('The above is not a grammatical '),
    hypothesis_formers(Lang,_),
    writelanguage(Lang,X),
    write(X),
    write(' sentence'),
    write('.'),
    nl.

```

```

/*=====
                                LEXICON
=====*/

%%% JAPANESE

lexicon(sukida,verb,stativeverb,[[_np1,ga],[_np2,ga]]).

lexicon(sinda,verb,intrverb,[[_np,ga]]).

lexicon(sinda-to,verb,intrverb,[[_np,ga]]).

lexicon(omotteiru,verb,matrixverb,[[_np,ga],[_Embedded]]).

lexicon(pn,taro).

lexicon(pn,jiro).

%%% DUTCH

lexicon(pn,jan).

lexicon(pn,piet).

lexicon(zwemmen,verb,_,[[_np,nom]]).

lexicon(zag,verb,_,[[_np,nom],[_Embedded]]).

/*=====
                                ABDUCTION RULES
=====*/

%%%abduce_from(+InputList)
%%%=====
%%%
%%%Abduce hypotheses based on InputList.

abduce_from(InputList):-
    hypothesis_formers(_,H),!,
    abducted_hyps(InputList,H,[],_).

```

```

%%%abduced_hyps(+InputList,+HypothesisFormers,
%%%          +WorkingMemory,-WorkingMemory)
%%%=====
%%%
%%% Update working memory after abducing hypotheses.

% Base case:

abduced_hyps(InputList, [], WM, WM) :-
    assertz(working_memory(WM)),
    write_to_log(InputList, WM),
    nl.

abduced_hyps(InputList, [hyp(Instance, Vtype, ArgsList) | T], TempWM, _) :-
    get_args(InputList, hyp(Instance, Vtype, ArgsList), ListofHyps),
    append(TempWM, ListofHyps, NewWM),
    abduced_hyps(InputList, T, NewWM, _).

% Recursively examine list of hypothesis formers for a match:

abduced_hyps(InputList, [_Failed | T], TempWM, _) :-
    abduced_hyps(InputList, T, TempWM, _).

% Simple case: hypothesis saturates input list

get_args(InputList, hyp(Instance, V, ArgsList),
          [hyp(Instance, V, ArgsList)]):-
    InputList = ArgsList.

% Complex case: hypothesis fails to saturate input and hypothesis
% is not a monoclausal structure, so do an incremental match:

%%%get_args(InputList, Hyp, WMList)
%%%=====
%%%
%%%
%%%

get_args(InputList, _, EmptyList) :- length(InputList, X),
                                     X =< 1,
                                     EmptyList = [].

```

```

get_args(InputList,hyp(Instance,matrixverb,ArgsList),ListofHyps):-
    incrementalmatch(InputList,hyp(Instance,matrixverb,ArgsList),
        ListofHyps).

% Base case: stop when no more input left:
incrementalmatch([],_,_).

% If first item in InputList matches first item in ArgsList,
% look up hypothesis formers and find a hypothesis that
% matches rest of input

incrementalmatch([H|T],hyp(Instance,Vtype,[H1|TArgs]),
    [hyp(Instance,Vtype,[H1|TArgs])]):-
    H = H1,
    hypothesis_formers(_,HF),
    search_hf(T,HF,TArgs).

search_hf(_T,[],_).

% Try first hypothesis in HF:

search_hf(RestInputList,[hyp(Instance,VerbType,Args)|_Tail],
    Result) :-
    RestInputList = Args,
    [[hyp(Instance,VerbType,Args)]] = Result.

% If first hyp fails to match, try next one:

search_hf(T,[_Failed|Tail],_Hypothesis):-
    search_hf(T,Tail,_Hypothesis).

write_to_log(InputList,WM):-
    nl,
    write('For input: '),
    write(InputList),nl,nl,
    write(' Hypotheses generated are: '),
    write(WM),!.

```

```

%%%matchverb(+Verb,+WorkingMemory)
%%%=====
%%%
%%% Try to match a verb with a hypothesis held in
%%% working memory. This predicate is called by Case 1 above.

% If first hypothesis is an embedded clause, do some
% special processing.
% Case 1: matrix verb in hypothesis matches Verb (Dutch case)
% This means that Verb is a matrix verb.

matchverb(Verb,[hyp(Verb,matrixverb,Args)|TailWM]):-
    lexicon(Verb,verb,matrixverb,_),
    retract(working_memory(_)),
    append([hyp(Verb,matrixverb,Args)],TailWM,NewWM),
    assertz(working_memory(NewWM)).

% Case 2: matrix verb in hypothesis does *not* match Verb;
% this is the usual Japanese/Hindi/German situation
% this means that Verb is not a matrix verb, e.g.,
% it could be an embedded verb, if the arguments match etc.
% Args might have an embedded hypothesis that matches the
% verb, so return a list Hyp containing the embedded
% hypothesis if there is one, then use matchverb3 to try
% to match Verb with the embedded hypothesis.
% If the match succeeds, update WM.

matchverb(Verb,[hyp(Verb1,matrixverb,Args)|TailWM]):-
    find_hyp(Args,[],Hyp),
    !,
    matchverb3(Verb,Hyp,Args),
    my_unify(Hyp,Args),
    retract(working_memory(_)),
    append([hyp(Verb1,matrixverb,Args)],
           TailWM,NewWM),
    assertz(working_memory(NewWM)).

```

```

% The unembedded case:

matchverb(Verb, [hyp(Verb, Type, Args) | _TailWM]):-
    lexicon(Verb, verb, Type, Args),
    retract(working_memory(_)),
    assertz(working_memory([hyp(Verb,
                                Type, Args)])).

% If first hypothesis fails, make a recursive call to rest of WM
% contents:

matchverb(Verb, [_Failedhyp|TailWM]):- matchverb(Verb, TailWM).

% If no hypothesis in WM matches, write an exception:

matchverb(_Verb, []):- write_exception.

% Base case: If there are no more embedded hypotheses, it failed
% to match the main hypothesis, so succeed vacuously.

matchverb3(_Verb, [], _):- true.

matchverb3(Verb, [hyp(Verb, Type, Args) | _TailWM], _):-
    lexicon(Verb, verb, Type, Args).

% If the above embedded hypothesis match fails, search deeper to
% see whether there is anything more. Recursive call to rest of
% embedded hypotheses:

matchverb3(Verb, [_FailedHyp|TailWM], _):- matchverb3(Verb, TailWM, _).

% If first element in ArgsList is a hypothesis, append it to a
% temp list and then look further in ArgsList.
% Recursion bottoms out when ArgsList empty.

find_hyp([], Hyps, Hyps):- true.

```

```

% If first argument is a hypothesis:

find_hyp([[Head] | _Tail], Temp, _Hyps):-
    Head = hyp(V,T,A),
    append(Temp, [hyp(V,T,A)], NewTemp),
    find_hyp([Head | _Tail], NewTemp, _).

% If first argument is not a hypothesis (could be an NP,
% i.e., a list of lists), search rest of Args:

find_hyp([_Failed | Tail], Temp, _Hyps):-
    find_hyp(Tail, Temp, _).

/*=====
                                UNIFICATION RULES
=====*/

my_unify([Hyp], [[hyp(_X,_Y,Args)] | _TailArgs]):-
    my_unify([Hyp], Args).

my_unify([Hyp], [[hyp(_X,_Y,Args)] | _TailArgs]):-
    Hyp = hyp(_X,_Y,Args).

my_unify(Hyp, [_HeadArgs | TailArgs]):- my_unify(Hyp, TailArgs).

```

## REFERENCES

- Abeillé, A., & Rambow, O. (Eds.). (2000). *Tree adjoining grammars*. Stanford, CA: CSLI.
- Abney, S., & Johnson, M. (1991). Memory requirements and local ambiguities of parsing strategies. *Journal of Psycholinguistic Research*, 20(3), 233–250.
- Aho, A. V., & Ullman, J. D. (1972). *The theory of parsing, translation and compiling, Vol.I: Parsing*. Englewood Cliffs: Prentice Hall.
- Aho, A. V., & Ullman, J. D. (1973). *The theory of parsing, translation and compiling, Vol.II: Compiling*. Englewood Cliffs: Prentice Hall.
- Aho, A. V., & Ullman, J. D. (Eds.). (1993). *Principles of compiler design*. New Delhi, India: Narosa Publishing House.
- Aissen, J. (2000). *Differential object marking: Iconicity vs. economy*. (MS. UCSC)
- Anderson, J. R., & Lebiere, C. (Eds.). (1998). *The Atomic Components of Thought*. Mahwah, New Jersey: Lawrence Erlbaum Associates.
- Babyonyshev, M., & Gibson, E. (1999). The complexity of nested structures in Japanese. *Language*, 75(3), 423–450.
- Bach, E., Brown, C., & Marslen-Wilson, W. (1986). Crossed and nested dependencies in German and Dutch: A psycholinguistic study. *Language and Cognitive Processes*, 1(4), 249–262.
- Bever, T. G. (1970). The cognitive basis for linguistic structures. In J. R. Hayes (Ed.), *Cognition and the development of language* (pp. 279–360). John Wiley.
- Bickel, B., & Yadava, Y. P. (2000). A fresh look at grammatical relations in Indo-Aryan. *Lingua*, 110, 343–373.
- Brants, T., & Crocker, M. (2000). Probabilistic parsing and psychological plausibility. In *Proceedings of 18th International Conference on Computational Linguistics COLING-2000*. Saarbrücken/Luxembourg/Nancy.
- Briscoe, E. (1987). *Modelling human speech comprehension: A computational approach*. Chichester: Ellis Horwood.

- Briscoe, E. J. (2000). Grammatical acquisition: Inductive bias and coevolution of language and the language acquisition device. *Language*, 76(2).
- Caplan, D., & Waters, G. S. (1999). Verbal working memory and sentence comprehension. *Behavioral and Brain Sciences*, 22, 77–94.
- Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.
- Chomsky, N., & Miller, G. (1963). Introduction to the formal analysis of natural languages. In R. D. Luce, R. R. Bush, & E. Galanter (Eds.), *Handbook of Mathematical Psychology, volume ii* (pp. 269–321). John Wiley.
- Clocksink, W. F., & Melish, C. S. (1981). *Programming in PROLOG*. Springer Verlag (Heidelberg, FRG and New York NY, USA).
- Console, L., Portinale, L., & Dupré, D. (1996). Using compiled knowledge to guide and focus abductive diagnosis. *Journal of IEEE Transactions on Knowledge and Data Engineering*, 8(5), 690–706.
- Covington, M. (1994). *Discontinuous dependency parsing of free and fixed word order*.
- Cussens, J., & Pulman, S. G. (2000). *Experiments in inductive chart parsing*. (ILPnet2 online library, available from: <http://www.cs.bris.ac.uk/~ILPnet2/>)
- Dickey, M. W., & Vonk, W. (1997). *Center-embedded structures in Dutch: An on-line study*. (MS)
- Evers, A. (1975). *The transformational cycle in Dutch and German*. Unpublished doctoral dissertation, University of Utrecht, The Netherlands.
- Frazier, L. (1979). *On comprehending sentences: Syntactic parsing strategies*. Unpublished doctoral dissertation, University of Massachusetts, Amherst.
- Gibson, E. (1998). Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68, 1–76.
- Gibson, E. (2000). Dependency locality theory: A distance-based theory of linguistic complexity. In A. Marantz, Y. Miyashita, & W. O’Neil (Eds.), *Image, language, brain: Papers from the first mind articulation project symposium*. Cambridge, MA: MIT Press.
- Hale, J. (2001). A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*.
- Harman, G. (1965). The inference to the best explanation. *Philosophical Review*, 74, 88–95.

- Hawkins, J. A. (1994). *A Performance theory of order and constituency*. New York: Cambridge University Press.
- Hawkins, J. A. (1998). Some issues in a performance theory of word order. In A. Siewierska (Ed.), *Constituent order in the languages of Europe*. Berlin: Mouton de Gruyter.
- Hepple, M. (1992). Chart parsing lambek grammars: Modal extensions and incrementality. In *Proceedings of COLING 1992*.
- Hepple, M. (1998). Memoisation for glue language deduction and categorial parsing. In *Proceedings of the COLING-ACL Joint Conference, Ninth International Congress on Linguistics*. Montreal, Canada: The Association for Computational Linguistics.
- Hobbs, J. R., Stickel, M. E., Appelt, D. E., & Martin, P. (1993). Interpretation as Abduction. *Artificial Intelligence*, 63, 69–142.
- Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory, languages and computation*. Reading, MA: Addison-Wesley.
- Hudson, R. (1996). The difficulty of (so-called) self-embedded structures. In P. Backley & J. Harris (Eds.), *UCL Working Papers in Linguistics* (Vol. 8(1), pp. 283–314).
- Johnson, N. F. (1977). A pattern-unit model of word identification. In D. LaBerge & S. J. Samuels (Eds.), *Basic processes in reading: Perception and comprehension* (pp. 91–125). Lawrence Erlbaum Associates.
- Johnson, N. F. (1981). Integration processes in word recognition. In O. Tzeng & H. Singer (Eds.), *Perception of print: Reading research in experimental psychology* (pp. 91–125). Lawrence Erlbaum Associates.
- Johnson, N. F. (1991). Holistic models of word recognition. In R. R. Hoffman & D. S. Palermo (Eds.), *Cognition and the symbolic processes* (pp. 79–93). Lawrence Erlbaum Associates.
- Johnson, N. F., Turner-Lyga, M., & Pettergrew, B. S. (1986). Part-whole relationships in the processing of small visual patterns. *Memory and Cognition*, 14, 5-16.
- Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness*. Cambridge, MA: Harvard University Press.
- Josephson, J. R., & Josephson, S. G. (1996). *Abductive Inference: Computation, Philosophy, Technology*. Cambridge, UK: Cambridge University Press.

- Joshi, A. (1996). Survey of the state of the art in human language technology. In G. B. Varile & A. Zampolli (Eds.), *Tree Adjoining Grammars* (pp. 406–411). Oregon: Center for Spoken Language Understanding, Oregon Graduate Institute.
- Joshi, A. K. (1990). Processing crossed and nested dependencies: An automaton perspective on the psycholinguistic results. *Language and Cognitive Processes*, 5(1), 1–27.
- Jurafsky, D. (1996). A probabilistic model of lexical and syntactic access and disambiguation. *Cognition*, 20, 137–194.
- Kaan, E., & Vasić, N. (2000). *Cross-serial dependencies in Dutch: Evidence for a discourse-based metric?* (Poster presented at CUNY Sentence Processing Conference, University of California, San Diego)
- Klabunde, R., & Jansche, M. (1998). Abductive reasoning for syntactic realization. In E. Hovy (Ed.), *Proceedings of the Ninth International Workshop on Natural Language Generation* (pp. 108–117). New Brunswick, New Jersey: Association for Computational Linguistics.
- Konieczny, L. (2000). Locality and parsing complexity. *Journal of Psycholinguistic Research*, 29(6), 627–645.
- König, E. (1990). The complexity of parsing with extended categorial grammars. In *Proceedings of COLING 1990*.
- Korthals, C. (2001). Self embedded relative clauses in a corpus of German newspaper texts. In K. Striegnitz (Ed.), *Proceedings of the Sixth ESSLLI Student Session* (pp. 179–190). Finland: University of Helsinki.
- Kruijff, G.-J. M. (1999). Resource logics, grammars, and processing. In C. Retoré & E. Stabler (Eds.), *Proceedings of the Workshop on Resource Logics and Minimalist Grammars*. Utrecht: ESSLLI.
- Kruijff, G.-J. M. (2001). *A Categorical-Modal Architecture of Informativity: Dependency Grammar Logic and Information Structure*. Unpublished doctoral dissertation, Charles University, Prague, Czech Republic.
- Lewis, R. L. (1996a). Interference in short-term memory: The magical number two (or three) in sentence processing. *Journal of Psycholinguistic Research*, 25(1), 93–115.
- Lewis, R. L. (1996b). *A theory of grammatical but unacceptable embeddings*. (Unpublished manuscript, Princeton University)

- Lewis, R. L., & Nakayama, M. (2001). Syntactic and positional similarity effects in the processing of Japanese embeddings. In M. Nakayama (Ed.), *Sentence Processing in East Asian Languages* (pp. 85–113). Stanford, CA.
- Marcus, M. P. (1980). *A theory of syntactic recognition for natural language*. Cambridge, MA: MIT Press.
- Miller, G., & Chomsky, N. (1963). Finitary models of language users. In R. D. Luce, R. R. Bush, & E. Galanter (Eds.), *Handbook of Mathematical Psychology, Volume II* (pp. 419–492). John Wiley.
- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*, 81–97.
- Moortgat, M. (1997). Categorical type logics. In J. van Benthem & A. ter Meulen (Eds.), *Handbook of logic and language*. Amsterdam: Elsevier.
- Nakatani, K., Babyonyshev, M., & Gibson, E. (2000). *The complexity of nested structures in Japanese*. (Poster presented at the CUNY Sentence Processing Conference, University of California, San Diego)
- Narayanan, S., & Jurafsky, D. (1998). Bayesian models of human sentence processing. In *Proceedings of the Cognitive Science Conference*.
- Pereira, F. C. (1985). A new characterization of attachment preferences. In D. R. Dowty, L. Karttunen, & A. M. Zwicky (Eds.), *Natural language parsing: Psychological, computational, and theoretical perspectives* (pp. 307–319). Cambridge University Press.
- Pollard, C., & Sag, I. A. (1994). *Head-driven phrase structure grammar*. Chicago: University of Chicago Press.
- Rambow, O., & Joshi, A. K. (1994). A processing model for free word order languages. In J. C. Clifton, C., L. Frazier, & K. Rayner (Eds.), *Perspectives on sentence processing*. Hillsdale, NJ: L. Erlbaum.
- Reich, P. A. (1969). The finiteness of natural language. *Language*, *45*(4), 831–843.
- Resnik, P. (1992). *Left-corner parsing and psychological plausibility*.
- Retoré, C., & Stabler, E. (to appear). Generative grammars in resource logics. *Journal of Language and Computation*, *3*.
- Roark, B. E. (2001). *Robust probabilistic predictive syntactic processing*. Unpublished doctoral dissertation, Brown University, Providence, RI.
- Roeck, A. D., Johnson, R., King, M., Rosner, M., Sampson, G., & Varile, N. (1982). A myth about center-embedding. *Lingua*, *58*, 327–340.

- Saint-Dizier, P. (1994). *Advanced logic programming for language processing*. London, UK: Academic Press.
- Sampson, G. (1996). From central embedding to corpus linguistics. In J. Thomas & M. Short (Eds.), *Using Corpora for Language Research*. Longman.
- Scheepers, C., Hemforth, B., & Konieczny, L. (1999). Incremental Processing of German Verb-Final Constructions: Predicting the Verb's Minimum (!) Valency. In *The Second International Conference on Cognitive Science, Japan*.
- Schlesewsky, M., Fanselow, G., & Kliegl, R. (no date). *The Costs of Wh-Movement in German*. (MS, University of Potsdam, available from <http://www.ling.uni-potsdam.de/~fanselow/cost.htm>)
- Shieber, S. (1986). *Introduction to unification-based approaches to grammar*. Stanford, CA: CSLI.
- Shieber, S. M. (1983). Sentence disambiguation by a shift-reduce parsing technique. In *International Joint Conference on Artificial Intelligence* (p. 699-703). Karlsruhe, Germany.
- Sikkel, K. (1997). *Parsing schemata*. Berlin: Springer.
- Smith, E. (2000). *Incoherence and text comprehension: Cognitive and computational models of inferential control*. Unpublished doctoral dissertation, University of Birmingham, UK.
- Steedman, M. (2000). *The Syntactic Process*. Cambridge, MA: MIT Press.
- Stolcke. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21, 165–202.
- Strigin, A. (1998). Lexical rules as hypotheses generators. *Journal of Semantics*, 15, 163–190.
- Strigin, A. (2001). *Constructing lexical meaning by hypothetical inference in context*. Unpublished doctoral dissertation, University of Berlin, Germany.
- Tyler, L. K., & Marslen-Wilson, W. D. (1977). The on-line effects of semantic context on syntactic processing. *Journal of Verbal Learning and Verbal Behavior*, 16, 683–692.
- Uehara, K., & Bradley, D. (1996). The effect of *-ga* sequences on processing Japanese multiply center-embedded sentences. In *11th Pacific-Asia Conference on Language, Information, and Computation*.

- Vasishth, S. (to appear, 2002). *Working memory in sentence comprehension: An empirical investigation of Hindi center embeddings*. Unpublished doctoral dissertation, Ohio State University, Columbus, OH.
- Vasishth, S., & Joseph, B. D. (2002). Constellations, Polysemy, and Hindi *-ko*. In *Proceedings of the Berkeley Linguistics Society Conference*. University of Berkeley, CA.
- Vermaat, W. (1999). *Controlling movement: Minimalism in a deductive perspective*. Unpublished master's thesis, University of Utrecht, The Netherlands.
- Warren, T., & Gibson, E. (1999). *The effects of discourse status on intuitive complexity: Implications for quantifying distance in a locality-based theory of linguistic complexity*. (Poster presented at the Twelfth CUNY Sentence Processing Conference, New York)
- Yamashita, H. (1997). The Effects of Word Order and Case Marking Information on the Processing of Japanese. *Journal of Psycholinguistic Research*, 26(2), 163–188.
- Yngve, V. H. (1960). A model and an hypothesis for language structure. *Proceedings of the American Philosophical Society*, 104, 444–466.