



UPPSALA  
UNIVERSITET

Department of Linguistics and Philology  
Uppsala University  
Språkteknologiprogrammet  
(Language Technology Programme)  
Master's thesis in Computational Linguistics  
17th March 2006

# Processing Dependencies

Working Memory Load in Sentence Parsing

Mattias Nilsson

Supervisor:  
Mats Dahllöf, Uppsala Universitet

## **Abstract**

This thesis presents a computational model of working memory load in human sentence processing that attempts to explain a number of well-established linguistic performance effects that are associated with processing difficulty of certain syntactic structures. The computational model presented here is based on recent assumptions of working memory in computational cognitive modeling. In particular the assumption that there are working memory demands in on-line sentence comprehension associated with both storage of incomplete syntactic structure and integration of new lexical input into the structure built thus far. The model has been realized as an implemented system for parsing a fragment of English into dependency representations. The implementation provides detailed quantitative predictions of the time course of sentence comprehension and processing difficulty. These predictions afford comparison with empirical evidence such as reading time data. In contrast to current sentence processing theories which rely on phrase structure representations, we implement an algorithm for parsing where the basic syntactic relations necessary for interpretation are established directly as asymmetrical dependency relations between lexical nodes. In addition to accounting for well-established processing contrasts, the dependency-based parsing model detailed here reflects important principles of real-time sentence processing, such as incrementality and predictivity.

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Contents</b>	<b>iii</b>
<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vi</b>
<b>Acknowledgments</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Cognitive modeling and language engineering . . . . .	2
1.2 Structure of the thesis . . . . .	3
<b>2 Working memory constraints in sentence processing</b>	<b>4</b>
2.1 Center-embedding . . . . .	4
2.2 Relative clause asymmetry . . . . .	6
2.3 Algorithms and acceptability bounds . . . . .	7
2.3.1 Top-down and bottom-up parsing . . . . .	8
2.3.2 The left corner algorithm . . . . .	8
2.4 Storage and computation . . . . .	9
2.5 The dependency locality theory . . . . .	10
2.5.1 Storage cost . . . . .	10
2.5.2 Integration cost . . . . .	11
2.5.3 The relationship between integration and storage cost . . . . .	13
<b>3 A dependency-based model for syntactic processing</b>	<b>14</b>
3.1 Dependency representations and dependency parsing . . . . .	14
3.2 Predictive processing . . . . .	17
3.2.1 Lexical constraints . . . . .	18
3.2.2 A predictive strategy . . . . .	19
3.2.3 ATTACH AS HEAD . . . . .	20
3.2.4 ATTACH AS DEPENDENT . . . . .	20
3.2.5 Parsing algorithm . . . . .	22
3.2.6 Parsing illustration . . . . .	23
3.3 Storage profiles . . . . .	23
3.3.1 <i>Right-branching structures</i> . . . . .	24
3.3.2 <i>Double center-embedding</i> . . . . .	25
3.3.3 <i>Subject- and object-relative clauses</i> . . . . .	25

<b>4</b>	<b>Working memory retrieval</b>	<b>28</b>
4.1	Discourse-based activation decay . . . . .	28
4.2	Retrieval profiles . . . . .	30
4.2.1	<i>Subject- and object-relative clauses</i> . . . . .	30
4.2.2	<i>Right-branching structures</i> . . . . .	32
4.2.3	<i>Double center-embedding</i> . . . . .	33
4.2.4	<i>Embedded pronominal subject in double center embedding</i> .	33
4.2.5	<i>Recency effects in structural ambiguity resolution</i> . . . . .	35
<b>5</b>	<b>Implementation</b>	<b>37</b>
<b>6</b>	<b>Summary</b>	<b>38</b>
	<b>Bibliography</b>	<b>40</b>

# List of Figures

3.1	Dependency graph for an English sentence . . . . .	15
3.2	Illustration of the word-by-word parse sequence for an English sentence	23
3.3	Dependency graph for the subject RC sentence in (26) . . . . .	26
3.4	Dependency graph for the object RC sentence in (27) . . . . .	26

# List of Tables

2.1	Word-by-word predictions of DLT storage cost for the sentence in (22) .	10
2.2	Word-by-word predictions of DLT integration cost for the subject-extracted RC structure in (24) . . . . .	12
2.3	Word-by-word predictions of DLT integration cost for the object-extracted RC structure in (25) . . . . .	13
3.1	Word-by-word predictions of storage cost for a right-branching structure	24
3.2	Word-by-word predictions of storage cost for a double center-embedded structure . . . . .	25
3.3	Word-by-word predictions of storage cost for the subject RC structure in (26) . . . . .	26
3.4	Word-by-word predictions of storage cost for the object RC structure in (27) . . . . .	27
4.1	Word-by-word predictions of retrieval cost for the subject RC structure in (28) . . . . .	30
4.2	Word-by-word predictions of retrieval cost for the object RC structure in (29) . . . . .	31
4.3	Word-by-word predictions of retrieval cost for the right-branching structure in (30) . . . . .	32
4.4	Word-by-word predictions of retrieval cost for the double center-embedded structure in (31) . . . . .	33
4.5	Word-by-word predictions of retrieval cost for the double center-embedded structure in (35) . . . . .	34
4.6	Word-by-word predictions of retrieval cost for the sentence in (36). Local attachment reading . . . . .	35
4.7	Word-by-word predictions of retrieval cost for the sentence in (36). Non-local attachment reading . . . . .	36

# Acknowledgments

I would like to express my sincere appreciation to Mats Dahllöf at the Department of Linguistics and Philology, Uppsala University, for supervising this thesis. I appreciate the continuous support and encouragement he has given me. I am also very grateful to Edward Gibson at the Department of Brain and Cognitive Sciences at Massachusetts Institute of Technology. I am thankful for his encouragement and for many valuable suggestions and helpful comments. I also want to thank Tessa Warren and Martin Pickering for their many valuable suggestions and comments. Finally, I would like to thank Joakim Nivre, Katja Suckow and Maria Chiara Benfatto for their encouragement and for reading and commenting on an earlier draft of this thesis.

# 1 Introduction

In order to understand a sentence in natural language it is necessary to assemble its syntactic structure and recover the semantic and referential representation associated with it. Syntactic structure assembly in human sentence comprehension requires some immediate memory of the past so that incomplete structure can be retained and integrated with new lexical input. Although words are input in succession and processed serially, linguistic dependencies may span several words. As a first example, consider the sentence in (1). This sentence involves a non-local subject-verb dependency spanning several words. The embedded relative clause *who my sister met by the sea last night* interrupts the dependency between the subject *the girl* and the main verb *wore*. Thus, when processing this sentence, some representation of the initial subject must be retained in memory before it can be syntactically and semantically integrated with the main verb in the structure.

(1) The girl who my sister met by the sea last night wore a yellow raincoat.

Experimental research has provided substantial evidence that sentence parsing requires processing resources in the form of working memory resources<sup>1</sup>. The evidence suggests that the efficiency of processing is constrained by inherent working memory limitations. One recent theory of the influence of working memory in sentence processing - the dependency locality theory (Gibson, 1998, 2000) - makes precise, quantitative predictions of processing load and hence processing difficulty during on-line sentence comprehension. According to the dependency locality theory, processing resources are required for two aspects of language comprehension: *storage* of the structure built thus far; and *integration* of new lexical input into the structure built thus far. Based on integration and storage, the dependency locality theory defines a structural metric over syntactic structures which can account for a rich variety of behavioral processing data across a range of languages.

In this thesis, we draw on the distinction between these two different functions of working memory in sentence processing. The aim of this work is to provide a computational model of working memory load in sentence processing, realized as an implemented model for parsing a fragment of English into dependency-based representations. The implemented model provides detailed quantitative predictions of word-by-word processing difficulty and on-line reading times. These predictions afford comparison with behavioral data from *self-paced reading time studies*, in which

---

<sup>1</sup>Research in cognitive psychology has justified a distinction between long-term memory, in which a large number of facts and autobiographical events are stored, and short term memory, in which very limited amounts of information are kept active for very brief periods of time. Short-term memory is generally referred to as a kind of working memory system: "... a short-duration limited-capacity memory system capable of simultaneously storing and manipulating information in the service of accomplishing a task" (Caplan and Waters, 1999).

participants are presented with sentences one word at a time and measurements of the time taken for the participant to request the next word are recorded<sup>2</sup>.

In contrast to current sentence processing theories which rely on phrase structure representations, we implement a model for parsing where the basic syntactic relations necessary for interpretation are established directly as asymmetrical dependency relations between lexical nodes. In addition to accounting for performance effects in a number of structures, the computational parsing model detailed here reflects important principles of real-time human sentence processing, such as incrementality and predictivity<sup>3</sup>.

## 1.1 Cognitive modeling and language engineering

The computational mechanisms which support human language processing form the central area of research in experimental and computational psycholinguistics. Understanding the mechanisms underlying language use and how they relate to general cognition is of critical importance to cognitive science in general. In addition, we argue that various aspects of linguistic performance also have implications for the engineering of natural language processing systems. In this view, computational theories and behavioral evidence can serve to inform research on language technologies. By way of illustration, consider the incremental nature of human sentence processing. Extensive experimental evidence has shown that language comprehension is incremental: each incoming word is processed immediately and integrated without delay into a partial sentence interpretation (e.g., Marslen-Wilson 1973; Just & Carpenter 1980; Frazier 1987). Many real-time NLP systems engaged in human communication must be able to reflect the immediacy of human sentence processing. For example, in order to build dialogue systems which can afford a natural dialogue flow, reaction times need to be reduced close to the order of that of humans. To achieve this, words need to be recognized and integrated into some structural representation as soon as they are encountered.

Turning to the matter at hand, any kind of information processing task makes cognitive demands on human processing capabilities. *Cognitive load* generally refers to the amount of information which must be retained in working memory in order to perform a given task. Information processing systems must respect the limitations of human working memory. If a system requires the user to process too much information at once, and thus to hold too many items in working memory, it will fail. This applies to NLP systems as well. Thus, a computational model which correctly predicts which kind of linguistic structures are associated with much cognitive effort in terms of processing load has a number of potential applications. For example, natural language generation systems, e.g., dialogue and machine translation systems,

---

<sup>2</sup> The typical procedure of a self-paced reading experiment is as follows. Each sentence is presented on a computer screen as a series of dashes which indicate the length and position of each word in the sentence. Participants press the spacebar to reveal each word in the sentence. Each subsequent press reveals the next word and removes the preceding word. The amount of time a participant spends reading each word is recorded as the time (in milliseconds) between key-presses. After the final word of each sentence, a comprehension question appears which prompts the participant to answer a yes or no question about the information in the previous sentence.

<sup>3</sup> This does not mean, however, that the implemented model should be considered a cognitive model in the strongly equivalent sense of Pylyshyn (1984). The processing steps in the implementation are not held to correspond to the processing steps of the human sentence processing system.

should preferably not produce output which people fail to understand or have difficulty in understanding. Filtering of sentences that are hard or difficult to process can be a useful component in such systems. A metric of processing load can also be used in such systems to select the syntactic structure associated with the least processing effort, so that the resulting sentence is easier to understand than other potential alternatives. Further applications include text readability, grammar and style checking software, and in general any computational linguistic application in which the comprehensibility of language is important. In addition, understanding the nature of working memory capacity limits have applications in any area concerned with cognitive overload in humans, such as aphasiology, attention-related disorders, multitask performance (e.g., driving while talking on the phone), the design of time-critical systems, user interface design etc.

However, before we consider efficient methods for reducing the working memory load in those NLP systems where the comprehensibility of language is particularly important, it is necessary to have basic computational models which make the accurate predictions with respect to well-established behavioral evidence. This is the task we address here.

## 1.2 Structure of the thesis

Throughout this thesis, we focus on those properties of the model that are relevant for understanding it as a computational model of sentence processing. Thus, the thesis does not address technical details of the implementation<sup>4</sup>. The remainder of this thesis is structured as follows. In chapter 2 we review some well-established experimental evidence for inherent working memory constraints in sentence processing. We also present a brief overview of the dependency locality theory. In chapter 3 we first consider general properties of dependency-based representations of syntactic structure and then detail an incremental and predictive strategy for parsing a fragment of English into such representations. A simple metric of working memory load, the number of open syntactic predictions, affords an account of a number of performance effects. In chapter 4 we focus on retrieval processes in sentence processing and the working memory resources required for performing structural integrations. We present a discourse-based activation decay model which correctly predicts a variety of performance phenomena. In Chapter 5 we provide some useful information about the implemented system. Chapter 6 provides a summary of the work presented in this thesis.

---

<sup>4</sup>The reader is referred to Chapter 5 for additional details on the implementation

## 2 Working memory constraints in sentence processing

Research over the past forty years has provided important insights into the nature of working memory and its influence on language processing. In this chapter we introduce a small set of linguistic structures for which there are substantial experimental evidence of particular processing effects arising as a consequence of working memory limitations. We also review some influential accounts of these performance effects. We will restrict our attention here, as well as in subsequent chapters, to unambiguous sentence processing difficulty. Thus, we will not be concerned with the processing difficulty associated with misinterpretation of local structural ambiguities, e.g. garden path effects<sup>1</sup>.

### 2.1 Center-embedding

One of the first clearly identified phenomenon associated with computational aspects of language in use was the difficulty in comprehending multiply center-embedded structures (Miller and Chomsky, 1963). A syntactic category, *X*, is center-embedded within another category *Y*, if there is some constituent in *Y* both to the left and to the right of *X*. Most native speakers of English find a sentence with a single center-embedded relative clause such as in (3) unproblematic to comprehend. In (3), the relative clause *that the cat bit* is center-embedded within the sentence *the mouse... ate the cheese*.

(3) The mouse that the cat bit ate the cheese.

There is a sharp drop in acceptability, however, from one to two or more levels of embedding in such structures. Thus, increasing the depth of embedding from one to two center-embedded relative clauses makes the sentence in (3) extremely difficult to process. This fact has been documented extensively in a large number of languages apart from English. The difficulty in processing English multiply center-embeddings is illustrated in (4) and (5) below<sup>2</sup>.

(4) # The mouse that the cat that the dog chased bit ate the cheese.

---

<sup>1</sup>A garden path effect arises as a consequence of a local structural ambiguity which is resolved in favour of an incorrect reading. The sentence in 2 provides a well-known example:

(2) The horse raced past the barn fell.

In (2) people usually misinterpret *raced* as a simple past tense verb, leading to processing breakdown, instead of as a passive participle, which is the correct interpretation: *the horse, which was raced past the barn, fell*.

<sup>2</sup>Following notational conventions we index *grammatical* but *unacceptable* sentences with #.

- (5) # The mouse that the cat that the dog that the elephant admired chased bit ate the cheese.

In (4), there is one relative clause embedded inside another relative clause. The relative clause *that the dog chased* is embedded between the subject *the cat* and the verb *bit* of the first relative clause. In (5), there are two relative clauses embedded inside another relative clause.

In contrast to center-embedded structures such as (4) and (5), structures that are right- or left-branching can be embedded essentially without limit. For example, the sentence in (6) is the right-branching counterpart of (4), and (7) is an example of a left-branching structure. Both of these sentence constructions are acceptable.

- (6) The dog chased the cat that bit the mouse that ate the cheese.

- (7) John's sister's doctor's cat hates mice.

The structural difference between these three constructions are schematically illustrated below:

- (8) *Right-branching*: [ $\alpha \dots [\alpha \dots]$ ]

- (9) *Left-branching*: [[ $\alpha \dots$ ]  $\alpha \dots$ ]

- (10) *Center-embedding*: [ $\alpha \dots [\alpha \dots] \dots$ ]

It is generally assumed that the difficulty associated with center-embedding is caused by an inherent constraint on working memory resources. Although there is a general agreement that working memory limitations play a fundamental role in explaining the difficulty, it is much less clear why exactly the processing system fails to construct a representation for such constructions. One general explanation relies on the number of nominal arguments which must be temporarily retained in working memory before either argument can be integrated with its associated verb. The fact that human working memory is bounded suggests that the processing system does not have access to the memory resources required in order to retain the nominal arguments in memory, and as a result it fails in representing the associated syntactic structure. This kind of processing breakdown associated with center-embedded structures is generally referred to as a *processing overload* effect.

However, much research on embedded structures since the early work by Miller and Chomsky (1963) has shown that the issue of embedding is considerably more complex than initially assumed. For example, not all multiply center-embedded structures are unacceptable, and contrary to common assumptions, some multiply center-embedded structures occur surprisingly frequently in language (Roeck et al., 1982). Roeck et al. (1982) collected examples of naturally occurring center-embeddings from English and German newspaper texts and showed that such structures are occasionally produced. Thus, besides right-branching and left-branching constructions, there are a variety of acceptable center-embedded constructions (Cowper, 1976; Gibson, 1991; Lewis, 1993). For example, consider the sentences in (11) and (12). These sentences are acceptable even though both involve multiply center-embedding.

- (11) That the food that John ordered tasted good pleased him.

- (12) The possibility that the man who I hired is incompetent worries me.

In (11), two sentences (*the food tasted good, John ordered*) are center-embedded within the main clause (*That... pleased him*). In (12), a relative clause (*who I hired*) is embedded inside a complement clause (*that the man is incompetent*). Although both of these sentences involve two levels of center-embedding they are still acceptable and do not cause the processing difficulty associated with the center-embedded structure in (4).

Increasing the number of embeddings in a center-embedded structure increases processing difficulty. However, increasing the *similarity* of the embedded arguments causes additional difficulty, and making the arguments more distinct in some way facilitates processing (Miller and Chomsky, 1963; Lewis, 1993, 1996). Consider the Japanese construction in (13) below (Lewis, 1993).

(13) *John-wa Bill-ni Mary-ga Sue-ni Bob-o syookai sita to it-ta.*

John-Top Bill-Dat Mary-Nom Sue-Dat Bob-Acc introduces say

“John said to Bill that Mary introduced Bob to Sue.”

This sentence contains five nominal arguments before any verb occurs, yet such constructions are acceptable in Japanese. The crucial difference between the Japanese construction and the English double center-embedded structure in (4) is that the nominal arguments in the Japanese sentence are marked for case. Processing the Japanese sentence requires retaining no more than two nominal arguments of any particular syntactic function in memory: at most two subjects, two indirect objects, and a direct object. In contrast, processing the center-embedded structure in (4), requires retaining three nominal arguments of the same syntactic function (subject) before any verb occurs. Lewis (1993, 1996) showed that a range of cross-linguistic data on center-embeddings can be explained as *similarity-based interference* effects in a capacity limited working memory. The central assumption here is that retaining incomplete syntactic constituents with the same syntactic function causes interference effects in working memory during processing, resulting in increased processing difficulty. Based on behavioral data, Lewis suggested that the working memory system could store no more than two items under the same syntactic relation at any point during the course of processing a sentence<sup>3</sup>.

## 2.2 Relative clause asymmetry

There is considerable experimental evidence showing that it is harder to process an object relative clause sentence such as (14) than a subject relative clause sentence such as (15) (e.g., Ford, 1983; King & Just, 1991).

(14) The reporter who the senator attacked admitted the error.

(15) The reporter who attacked the senator admitted the error.

Even though people generally are able to comprehend object relative clauses, such structures are processed more slowly than subject relatives. This contrast has been

---

<sup>3</sup>Lewis original model has since been substantially revised. Lewis and Vasishth (2005) provides a detailed sentence processing theory based on activation decay and similarity-based interference in working memory.

established in several studies, using a variety of experimental techniques, e.g., self-paced reading time and event-related potentials<sup>4</sup>. The critical differentiating region occurs at the verb of the relative clause (*attacked*) which is associated with increased reading time in object relative clause sentences. Moreover, it has been reported that aphasic patients have great difficulty in comprehending object relative clauses (e.g., Grodzinsky, 1989).

In a transformational grammar framework that assumes the existence of empty categories, the syntactic structure of these different structures are accounted for as follows. The relative pronoun, *who*, in an embedded object relative clause, is initially extracted from the object position of the embedded clause. This empty category position is then co-indexed with the extracted relative pronoun:

(16) The reporter<sub>*i*</sub> [<sub>*S'*</sub> *who*<sub>*i*</sub> [<sub>*S*</sub> the senator attacked *e<sub>i</sub>*]] admitted the error.

Correspondingly, in an embedded subject relative clause the relative pronoun is extracted from the subject position of the embedded clause:

(17) The reporter<sub>*i*</sub> [<sub>*S'*</sub> *who*<sub>*i*</sub> [<sub>*S*</sub> *e<sub>i</sub>* attacked the senator]] admitted the error.

These structures are often referred to as object-extracted relative clauses and subject-extracted relative clauses, respectively.

The processing asymmetry between the two different structures, e.g., as measured by reading-times, has been given different explanations. Under a capacity-limited view, when processing the object-relative clause, two nominal arguments must be retained in working memory before any verbal information clarifies the sentence structure. By contrast, in the subject relative clause only one nominal argument precedes the embedded verb. Thus, processing load at the embedded verb is higher in the object relative clause, and as a result reading times increase at this word. A different influential account which will be reviewed in section 2.5 relies on the greater linear distance between the embedded verb and the extracted relative pronoun (or more precisely, between the empty category position *e<sub>i</sub>* and the relative pronoun) in the object relative clause. This account explains the observed contrast under the assumption that it is harder to integrate new lexical input with distant linguistic material than more recent material.

## 2.3 Algorithms and acceptability bounds

While there is no severe bound on the number of acceptable embeddings in right- or left-branching structures, center embedded relative clause structures are critically restricted, as previously discussed. There have been attempts to characterize human memory limitations in syntactic processing by considering these acceptability bounds relative to the computational memory requirements of particular parsing algorithms for context free grammars (Miller and Chomsky, 1963; Johnson-Laird, 1983; Abney and Johnson, 1991; Resnik, 1992). These accounts have relied on the stack size, i.e. the number of incomplete nodes at each point in the parsing process, as a reasonable metric for working memory load.

---

<sup>4</sup>*Event-related potentials (ERP)*: A measurement of brain activity which indicates the presence of syntactic (P-600) and semantic anomaly (N-400) during language comprehension.

### 2.3.1 Top-down and bottom-up parsing

In a strict top-down parsing algorithm for context free grammars the stack is maintained in order to keep track of syntactic categories that have been predicted thus far and therefore still need to be found. A top-down algorithm may require an unbounded amount of stack space in order to parse left-branching and center-embedded structures (Miller and Chomsky, 1963; Abney and Johnson, 1991). Top-down processing associates bounded (constant) space with right-branching structures only. Thus, a structural metric of processing load based on the memory requirements of a top-down algorithm suggests that left-branching structures should be as unacceptable as center-embedded structures and harder to process than right-branching structures. Since left-branching structures such as (18) are acceptable, such an assumption is incorrect with respect to behavioral data (Chomsky and Miller, 1963; Miller and Chomsky, 1963; Johnson-Laird, 1983; Abney and Johnson, 1991).

(18) John's sister's doctor's cat hates mice.

A similar analysis holds for a bottom-up algorithm. In bottom-up parsing the stack is maintained in order to keep track of categories that have been found thus far, and still need to be structured together. A bottom-up algorithm may require an unbounded amount of stack space in order to process right-branching and center-embedded structures (Chomsky & Miller, 1963; Abney and Johnson, 1991). Bottom-up processing associates constant space only with left-branching structures. Hence, a metric based on the memory requirements of a bottom-up algorithm suggests that right-branching structures should be as unacceptable as center-embedded structures and harder to process than left-branching structures. Since right-branching structures such as (19) are acceptable, such an assumption is also incorrect with respect to performance data (Chomsky and Miller, 1963; Miller and Chomsky, 1963; Johnson-Laird, 1983; Abney and Johnson, 1991).

(19) The dog chased the cat that bit the mouse that ate the cheese.

Taken together, these results suggest that the human sentence processing system does not function entirely top-down or entirely bottom-up.

### 2.3.2 The left corner algorithm

Neither standard top-down or bottom-up algorithms associate unbounded memory with center-embedded structures only. It has been shown, however, that an instance of the left corner algorithm do require unbounded stack space for center-embedded structures while retaining bounded space for both left- and right-branching structures (Aho and Ullman, 1973; Johnson-Laird, 1983; Abney and Johnson, 1991; Resnik, 1992). A left corner algorithm proceeds partially top-down and partially bottom-up by identifying the left most corner of the right hand side of a context-free rule bottom-up, and predicting the remaining categories on the right hand side of the rule top-down. Thus, when given a context free rule such as (20) below and a structure of category  $R_1$  is found bottom-up, a structure of category  $L$  is built above  $R_1$  and the categories  $R_2$  to  $R_n$  are predicted to the right. Category  $L$  is complete when the categories  $R_2$  to  $R_n$  have been found. At that point, category  $L$  may cause left-corner prediction for all those rules in the grammar in which  $L$  is the leftmost category of the right hand side.

$$(20) L \rightarrow R_1 R_2 \dots R_n$$

Left-Corner parsing defines a range of possible parsers depending on which arc-enumeration<sup>5</sup> strategy that is employed. For example, an *arc-eager* strategy enumerates the arc between two nodes as soon as both nodes are present, while an *arc-standard* strategy enumerates the arc between two nodes only after all the nodes dominated by the child have been enumerated. While an arc-eager left corner parsing algorithm requires only constant space on left- or right-branching structures it may require an unbounded amount of memory in order to process center-embedded structures. Thus, the memory requirements of the arc-eager left corner algorithm correspond appropriately to acceptability bounds on processing with respect to left-, right- and center-embedded structures. However, it is clear that the diverse nature of human sentence processing difficulty can not be accounted for with reference to the memory requirements of the left corner algorithm alone.

## 2.4 Storage and computation

Most theories of the influence of working memory in language processing have relied on the conception of working memory as a limited-capacity buffer reserved for short-term storage of incomplete structure. For example, the algorithmic-based approach previously reviewed quantified processing load in terms of the number of nodes temporarily retained on the stack during parsing. However, much experimental evidence in more recent memory research has motivated a different conception of working memory. Thus, it is currently assumed that memory resources are required not only for short-term storage of the intermediate products of computations, but also for supporting the actual computations themselves. Working memory is better construed as both a storage space and an arena for computation (Just and Carpenter, 1992). For example, when performing a task such as keeping track of a list of unrelated words or numbers for a short period of time it is necessary to hold information in temporary storage to complete the task. However, when performing a task such as an arithmetic calculation, such as adding two three digit numbers, additional processing resources are required in order to compute the answer. Just and Carpenter (1992) suggested that sentence processing relies on both of these functions of working memory:

Working memory plays a critical role in storing the intermediate and final products of a reader's or listener's computations as she or he constructs and integrates ideas from the stream of successive words in a text or spoken discourse. In addition to its role in storage, working memory can also be viewed as the pool of operational resources that perform the symbolic computations and thereby generate the intermediate and final products.

The distinction between storage and computational resources has been particularly influential in one recent theory of working memory in sentence processing. Next, we present a brief overview of this theory.

---

<sup>5</sup>The order in which the nodes and arcs of the parse tree is enumerated (Abney and Johnson, 1991).

## 2.5 The dependency locality theory

The dependency locality theory (DLT, Gibson, 1998, 2000) instantiates the distinction between different functions for storage and computational resources in sentence comprehension. According to the DLT, working memory resources are required for two aspects of sentence comprehension: *storage* of the structure built thus far; and *structural integration* of new lexical input into the structure built thus far. Based on these two components, the DLT defines a structural metric over syntactic structures which predicts a wide range of processing effects in both unambiguous and ambiguous structures. Storage is associated with a cost measured in *memory units* (MUs), and structural integration is associated with a cost measured in *energy units* (EUs).

### 2.5.1 Storage cost

The processing resources required for storing a partially processed structure are proportional to the number of incomplete syntactic dependencies at that point in processing the structure. As a result, processing load increases when the number of incomplete dependencies increases. More specifically, storage cost is computed as follows. Each syntactic head required to complete the current input string as a grammatical sentence is associated with a storage cost of 1 MU. It is assumed that the minimal number of syntactic heads needed to complete a grammatical sentence is two: a noun for the subject, and a verb for the predicate. If words are encountered during processing that necessitate other syntactic heads to form a grammatical sentence, then these heads are also predicted and additional storage cost is incurred. For example, after processing the partial input string *the girl who ...*, in (21) below, there is a storage cost associated with syntactic expectations for two verbs, which are minimally required in order to form a grammatical sentence. The subject *the girl* requires a verb and the relative pronoun *who* also requires a verb. Thus, for example, the sentence can be completed grammatically as *the girl who sleeps shivers*.

(21) The girl who sleeps shivers

Consider the simple sentence in (22) and its storage cost profile presented in table 2.1.

(22) The reporter attacked the senator.

**Table 2.1:** Word-by-word predictions of DLT storage cost for the sentence in (22)

Syntactic prediction	Input word				
	The	reporter	attacked	the	senator
Matrix verb	1		R		
Subject	1	R			
Object NP			1		R
Total cost	2	1	1	1	0

For the first word, the determiner *the*, two syntactic heads are needed to complete a grammatical sentence, a noun for the subject and a verb for the predicate. Thus the storage cost is 2 MUs at this point. At the next word, *reporter*, the prediction of a

noun is realized and thus only a verb is required to complete the sentence. Hence the storage cost is 1 MU at this point. At the third word, *attacked*, a direct object is needed to complete the sentence grammatically, so the cost at this word is 1 MU. The following determiner *the* does not satisfy the prediction for an object, but it does not add any additional predictions either, so the storage cost remains at 1 MU here. The last word *senator* realizes the prediction of an object and completes the sentence grammatically.

The DLT Storage cost component can account for processing overload effects in multiply center embedded structures, as well as for the processing contrast between object relative clauses and subject relative clauses. In particular, at the point of having processed the partial input string *the man who the . . .*, in the previously discussed object relative clause structure, repeated here in (23), there are four predictions which must be satisfied in order to complete the sentence grammatically.

(23) The man who the senator attacked admitted the error

These predictions correspond to a main verb for the sentence, a verb for the relative clause, an empty category position in the RC (relative clause) to be associated with the relative pronoun *who*, and a noun for the determiner *the*. In contrast, when processing the subject relative clause in (15) there are never more than two open predictions at any given point.

Both storage cost and integration cost provide explanations for a wide range of performance effects. According to the DLT, however, integration cost reflects reading time effects most directly.

## 2.5.2 Integration cost

The structural integration cost associated with integrating a new word into the current syntactic structure increases as the linear distance increases between the word and the most local head or dependent to which it attaches in the structure. More precisely, the cost of integrating the current word  $w$  with a previous syntactic item is proportional to the number of discourse referents<sup>6</sup> that separates  $w$  and the associated syntactic item in the structure built thus far. The motivation for quantifying head-dependent distance in terms of discourse referents is based on experimental evidence which suggests that the difficulty of processing a nominal argument depends on the accessibility of the nominal referent in the discourse. It is harder to access or mentally construct referents which refer to new discourse referents than entities or individuals which are already in focus in the discourse. Focused entities or individuals, which are usually referred to with pronouns, are highly accessible. Nonfocused entities or individuals, usually referred to with proper names and definite descriptions, and indefinite noun phrases are less accessible and require the most memory resources:

The less accessible the referent of an NP is in the discourse, the more resources are required to find or construct it. . . . In particular, it is as-

---

<sup>6</sup> A discourse referent, according to the DLT, is 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 NP’s, or tense on a verb for events” (Gibson, 2000, 103).

**Table 2.2:** Word-by-word predictions of DLT integration cost for the subject-extracted RC structure in (24)

Cost type	Input word								
	The	reporter	who	attacked	the	senator	admitted	the	error
New discourse referent	0	1	0	1	0	1	1	0	1
Structural integration	0	0	0	0	0	0	2	0	0
Total	0	1	0	1	0	1	3	0	1

sumed that processing the head noun of an NP that refers to a new discourse object consumes substantial resources, and processing the head verb of a VP that refers to a new discourse event (also a discourse referent) consumes substantial resources, but processing other words does not consume substantial resources in the discourse processing component of structure building. (Gibson, 2000, 103).

The total integration cost at a word is the sum of all structural integrations taking place at that word. Consider the structural integration cost metric with respect to the subject-object relative clause asymmetry. The subject relative clause sentence is repeated here in (24) and its associated structural integration cost profile is presented in table 2.2.

(24) The reporter<sub>i</sub> [<sub>S</sub> who<sub>i</sub> [<sub>S</sub> e<sub>i</sub> attacked the senator]] admitted the error.

In the subject-relative clause, all integrations taking place are local until the main verb *admitted* is processed. Each of the previous words *reporter*, *attacked* and *senator* are associated with a discourse cost of 1 EU, since each of these words introduce new referents in the discourse. The word *admitted* is attached as the main verb for the subject *the reporter*. There is a discourse cost of 1 EU at the main verb corresponding to the referent *admitted*. In addition, there is a structural integration cost associated with integrating the main verb with the previous subject. The integration crosses two discourse referents, *attacked* and *senator*. Thus, the structural integration cost at the main verb word is 2 EUs.

Consider now the object relative clause sentence, repeated here in (25), and its associated integration cost profile given in table 2.3.

(25) The reporter<sub>i</sub> [who<sub>i</sub> [the senator attacked e<sub>i</sub>]] admitted the error.

The structural integration cost in the object relative clause differ from the subject relative clause only at the embedded verb *attacked*. At this word, there are three structural integrations taking place. The word *attacked* is attached locally as the verb for the subject *the senator* (0 EUs). An empty category, *e*, is integrated locally as the object of *attacked* (0 EUs). The empty category, *e*, is co-indexed with the pronoun *who*. This is a non-local integration. There are two discourse referents, *senator* and *attacked*, in the intervening region. Hence, the structural integration cost at the embedded verb is 2 EUs. The total integration cost at *attacked* is 3 EUs.

In sum, structural integration cost predicts the well-established increase in reading time at the embedded verb in an object relative clause. Processing this word requires more integration resources in an object relative clause than in a subject relative clause.

**Table 2.3:** Word-by-word predictions of DLT integration cost for the object-extracted RC structure in (25)

Cost type	Input word								
	The	reporter	who	the	senator	attacked	admitted	the	error
New discourse referent	0	1	0	0	1	1	1	0	1
Structural integration	0	0	0	0	0	2	2	0	0
Total	0	1	0	0	1	3	3	0	1

In addition to accounting for the lower complexity of subject-embedded relative clauses compared to object-extracted relative clauses, the DLT accounts for various processing overload effects, including the difficulty with multiple center-embeddings, the acceptability of right- and left-branching structures, length/heaviness effects<sup>7</sup>, various ambiguity effects and a number of other performance effects not discussed here.

### 2.5.3 The relationship between integration and storage cost

The DLT assumptions regarding the relationship between syntactic integration and syntactic storage are as follows: (i) structural integration and storage access the same pool of memory resources; (ii) there is a fixed capacity of processing resources; (iii) each predicted syntactic head takes up a fixed quantity of resources; and (iv) the overall acceptability of a sentence is reflected by the maximal integration and storage cost incurred during the parsing process.

Apart from storage cost and integration cost, there are additional factors assumed to influence the comprehensibility of a sentence: (i) the frequency of the lexical items being integrated, i.e. processing high-frequency lexical items require fewer resources than the processing of low-frequency lexical items; and (ii) the contextual plausibility of the resulting structure, i.e. structures representing plausible meanings require fewer resources than structures representing implausible meanings. The dependency locality theory, however, makes no further assumptions with respect to these factors.

---

<sup>7</sup> Length/heaviness effects refer to the fact that linguistic constructions are generally easier to process if the longer or “heavier” items occur later in the sentence.

## 3 A dependency-based model for syntactic processing

In this chapter, we detail a model for parsing a fragment of English into dependency-based representations. The parsing model presented here involves both bottom-up processing and top-down prediction of expected syntactic relations between words already seen and yet unseen words. A simple metric of working memory load, the number of syntactic predictions associated with incomplete dependencies, affords an account of a number of well-established performance contrasts. Before we turn to a description of the parsing model, we first consider some general properties of dependency representations and dependency parsing.

### 3.1 Dependency representations and dependency parsing

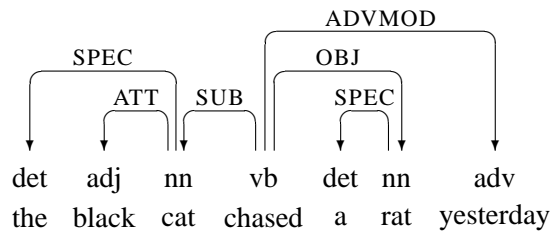
Dependency-based theories of syntax define the syntactic structure of sentences in terms of binary asymmetrical dependencies between the words of a sentence<sup>1</sup>. Thus, in contrast to syntactic representations based on constituency or phrase structure, there are no higher nodes (i.e. phrasal nodes) in dependency structures. A dependency relation holds between two terminal nodes, of which one is the head and the other is the dependent. Every lexical node in a dependency structure is dependent on at most one other lexical node. There is only one node which is not dependent on any other node. The node which is not dependent on any other is generally referred to as the root of the sentence. A dependency structure can be defined as a directed graph, where the set of nodes is given by the set of lexical elements, and the set of arcs represent dependency relations from heads to dependents (Nivre, 2005). Figure 3.1 shows a dependency structure for an English sentence represented as a directed graph, where each word of the sentence is labeled with its lexical category and each arc labeled with a syntactic function.

Given that dependency structures can be represented as directed graphs, conditions governing the well-formedness of such structures can be stated as graph constraints<sup>2</sup>. Thus, for example a dependency graph must be *connected* in order to provide a complete syntactic analysis of a sentence. This constraint ensures that each node is related to at least one other node. In addition, there are two more basic constraints generally imposed on dependency graphs: the *single-head* constraint and the

---

<sup>1</sup>For a short review of modern dependency grammar and dependency parsing, the reader is referred to Nivre (2005).

<sup>2</sup>For a more detailed discussion of dependency graphs and graph constraints, the reader is referred to Nivre (2003).



**Figure 3.1:** Dependency graph for an English sentence

*acyclicity* constraint. The single-head constraint ensures that each node is dependent on at most one other node, while the acyclicity constraint ensures that the graph does not contain cycles<sup>3</sup>. In addition to these constraints, a dependency graph may be *projective* with respect to the linear order of the words in the sentence. Informally, the projectivity constraint ensures that if a node  $h$  is the head of another node  $d$ , then each word  $w$ , which occurs between  $h$  and  $d$  in the linear order, is a direct or indirect dependent of  $h$ . By assuming the projectivity constraint, dependency graphs with crossing branches are rejected. Thus, there is a general acceptance in most dependency-based theories to recognize non-projective structures in order to account for discontinuous constructions, and the syntactic structure of languages with free or variable word order.

It should be noted that although not explicitly expressed, phrase structure or constituency is still recognized in dependency representations. A phrase or constituent in a dependency representation consists of any head and all its direct and indirect dependents. Thus, phrase structure can be derived from the binary dependency relations. The basic syntactic primitives in dependency representations, however, are the asymmetrical dependencies that hold between lexical items. These relations impose a hierarchically organized structure on the words of a sentence, such that the complete syntactic structure can be defined as a directed graph (or rooted tree).

In order to account for syntactic structure in terms of dependency relations, it is necessary to establish criteria which distinguishes the head from dependent in such relations. Dependency relations are generally motivated by both semantic and syntactic criteria. Some of the criteria which have been proposed for identifying a syntactic dependency relation between a head  $H$  and a dependent  $D$  in a construction  $C$  is presented below (Nivre, 2005).

1.  $H$  determines the syntactic category of  $C$  and can often replace  $C$ .
2.  $H$  determines the semantic category of  $C$ ;  $D$  gives semantic specification.
3.  $H$  is obligatory;  $D$  may be optional.
4.  $H$  selects  $D$  and determines whether  $D$  is obligatory or optional.
5. The form of  $D$  depends on  $H$  (agreement or government).
6. The linear position of  $D$  is specified with reference to  $H$ .

Mel'čuk (1988) makes a distinction between different linguistic levels of dependency relations. In particular, he recognizes three different types of dependencies: *morpho-*

<sup>3</sup>There are, however, dependency frameworks that do not impose the single-head or acyclicity constraint on the syntactic structures, e.g. *Word Grammar* (Hudson, 1984).

*logical, syntactic and semantic*. According to Mel'čuk, these three different kinds of dependencies are distinct in the structure of sentences and should not be confused.

Dependency representations are generally more constrained and hence less expressive with respect to certain aspects of syntactic structure than constituency-based representations (Nivre, 2005). However, while constituency representations are more expressive, the more constrained dependency representations present potential advantages with respect to the parsing of natural language into such representations (Nivre, 2005). Below, we summarize some of the arguments which have been put forward in support of parsing with dependency-based representations (based on Covington (2001) and Nivre (2005)).

- *Constrained parsing problem*  
In contrast to constituency-based representations, the number of nodes in a dependency structure is fixed by the input string itself. Thus, there is no postulation of additional nodes in dependency parsing. This affords conceptually simpler methods for parsing.
- *Correspondence to semantic interpretation*  
While being more constrained (and hence less expressive) than constituency-based representations, dependency representations provide a relatively direct encoding of predicate-argument structure. Thus, there is generally a direct correspondence of dependency relations to the structural relations that are necessary for semantic interpretation.
- *Immediacy of processing*  
Parsing with dependency-based representations affords immediacy of processing, that is, each word can (generally) be immediately integrated into the evolving representation after its occurrence in the input stream.
- *Accounting for discontinuity*  
Given that non-projective dependency structures are allowed, dependency-based parsing affords a satisfactory treatment of languages with variable word order, where discontinuous syntactic constructions are more common than in languages like English.

Covington (2001) presents variations on what is referred to as a fundamental algorithm for parsing natural language into dependency representations. His approach is based on a parsing algorithm originally formulated for discontinuous dependency parsing. Here, free word order is considered the most general case and restrictions on word order are realized by imposing additional constraints on the parsing strategy. The basic strategy is similar to bottom-up shift-reduce parsing for context free grammars. The parsing strategy presented in Covington (2001) is deterministic, in the sense that once a dependency relation has been established it is assumed to be part of the final structure and can not be removed. However, in order to handle local ambiguity, Covington recognizes a limited backtracking mechanism. The time complexity of the deterministic parsing algorithm is  $O(n^2)$ . The fundamental parsing strategy assumes only a dependency grammar defined in terms of binary dependency rules. Given any two words, the grammar should be able to determine whether a dependency relation can be established between these words and to identify which is the head and which is the dependent. The fundamental left-to-right strategy is as follows (Covington, 2001):

Accept words one by one starting at the beginning of the sentence, and try linking each word as head or dependent of every previous word.

This strategy does not explicitly state in which order a head or dependent is searched for. Thus, when searching for a previous head or dependent of the current word  $w_i$ , it is possible to start either from the most recently processed word  $w_{i-1}$ , or from the beginning of the input string. However, given that syntactic relations tend to be local, it is concluded that the former strategy is likely to be more efficient. The most-recent-first search strategy is subsequently refined by imposing different constraints on the linking operation. Thus, the single-head (uniqueness) and projectivity constraint are built into the parsing algorithm. The single-head constraint is implemented as follows: (i) when looking for dependents of the current word, do not consider words that are already dependents of something else; (ii) when looking for the head of the current word, stop after finding one head, there will not be another. The projectivity constraint is implemented as follows: (i) do not skip a potential predependent of the current word, that is, either attach every consecutive preceding word that is still independent, or stop searching; (ii) when searching for the head of the current word, consider only the previous word, its head, that word's head, and so on to the root of the tree. In summary, the algorithm for dependency parsing presented in Covington (2001) provides a way to accept free word order while at the same time maintaining a recency preference for syntactic attachments. In addition, by modifying the linking operation, this parsing strategy can be implemented so as to produce dependency structures which satisfy the constraints of *single-head* and *projectivity*.

## 3.2 Predictive processing

Recent psycholinguistic evidence suggests that language comprehension is *anticipatory* or *predictive*. People not only process language incrementally, but also make continuous predictions about what they expect to follow next (e.g., Kamide *et al.*, 2003a; Kamide *et al.*, 2003b). Thus, in addition to providing sufficient information for attaching the word being processed into the existing structure without delay, previously processed input also provides cues to linguistic properties of subsequent input. As previously shown, retaining predictions for incomplete dependencies is associated with processing load in the storage cost component of the dependency locality theory<sup>4</sup>. Accordingly, the DLT predicts that processing difficulty should increase when the number of syntactic predictions increases in the structure.

Most current approaches to dependency parsing do not involve anticipatory or predictive (*top-down*) processing. For example, the parsing algorithm in Covington (2001) reviewed above, proceeds by accepting words from left to right, first trying to attach the current word as the head of previous words and then to attach the current word as a dependent of a previous word. There is, however, no attempt to predict any syntactic properties of subsequent heads or dependents following the current word. Thus, if there is no previous node to which the current word  $w$  can attach as dependent, no further attempt is made to identify a subsequent head node for  $w$ , although the possible syntactic identities of such a head may be constrained by the lexical properties of  $w$ . Correspondingly, there is no attempt to identify a subsequent

---

<sup>4</sup>More precisely, each predicted syntactic head required to complete the current input string as a grammatical sentence is associated with increased working memory load in the DLT.

dependent node for the current word  $w$ , although the dependent may be required and its syntactic properties constrained by the properties of  $w$ .

Here, we will detail a preliminary attempt at a parsing strategy which involves top-down prediction of expected syntactic relations between words already seen and yet unseen words. We present a parsing algorithm based on Covington (2001) which embodies both bottom-up and top-down processing. Although the parsing strategy has been realized as an implemented parser, it must be emphasized that its performance has been assessed only with respect to the particular syntactic structures it was designed to account for. Additional functions will need to be added in order to parse a larger fragment of English appropriately.

### 3.2.1 Lexical constraints

The syntactic attachments considered by the parser are restricted by the lexical constraints associated with each lexical item. These constraints apply during the processing of a sentence as lexically-based predictions about the categories that can follow the current lexical item  $w$ , which are in a head-dependent relationship with  $w$ . Given that a dependent node generally presupposes the presence of the head and that a head may require the presence of the dependent, anticipated nodes may be predictions of either head nodes or dependent nodes. Parsing, as construed here, is a process of looking up the lexical features of the current word and trying to match these features to the predictions in the dependency structure built thus far. We will assume a dependency grammar formalism similar to Covington (2001). The formalism employed here differs, however, from Covington (2001) in that it defines *directed* dependency rules (Nivre, 2003) and distinguishes between *complements* and *modifiers* of a head. A directed dependency rule identifies the relative position, in the linear order, of the dependent with respect to the head. Thus, a dependency relation in which the dependent precedes its head in surface syntax is *left-directed*, while a dependency relation in which the dependent follows its head is *right-directed*. The distinction between complements and modifiers (or adjuncts) is central to syntactic theory. A lexical head may impose *valency* requirements on its syntactic dependents. In particular, the lexical properties of a verbal head identifies *complements* of the verb which are either required or syntactically permitted by the verb. Thus, complements are valency-bound by their head. By contrast, dependents that are not valency-bound by their head are modifiers. Modifiers are generally optional dependents of the head.

In the current formalism, each word is defined by a lexical entry,  $L_w$ , that specifies the orthographic form of the word, lexical category information and agreement features of the word<sup>5</sup>(e.g., *person, number, gender case, tense*). In addition, lexical entries are anchored to lexical constraints, which are defined in terms of a set of dependency rules.

We define a dependency grammar as a quintuple  $\langle W, C, D, F, R \rangle$ , where

- $W$  is a vocabulary; a finite set of word forms;
- $C$  is a finite set of lexical categories;
- $D$  is a finite set of dependency relations, e.g., *subject, object, adverbial*;

---

<sup>5</sup>For expository purposes, the internal agreement features of words are not explicitly defined in the subsequent definition of the formalism.

- F is an assignment function ( $F : W \rightarrow C$ );
- R is a set of dependency rules of the following form:

$\langle [\text{HEAD}, \text{COMPLEMENT}, \text{MODIFIER}, \text{DIRECTION}] \rangle$ , where

HEAD is the head of the rule; a lexical category  $C_h \in C$ ;

COMPLEMENT identifies complements of HEAD in terms of a set of feature structures of the form:  $[funct: Funct, cat: C_d]$ , where  $Funct \in D$  and  $C_d \in C$ ;

MODIFIER identifies a modifier of HEAD in terms of a feature structure of the form:  $[funct: Funct, cat: C_d]$ , where  $Funct \in D$  and  $C_d \in C$ ;

DIRECTION identifies the linear position of COMPLEMENT and MODIFIER with respect to HEAD, i.e., either *left* or *right*;

Thus, *left*-directed rules identify valency-bound and optional dependents to the left of the head, while *right*-directed rules identify valency-bound and optional dependents to the right of the head.

### 3.2.2 A predictive strategy

The general principle which drives the parsing process is as follows:

Accept words one by one from left to right, and try attaching each word as head or dependent of every previous word by matching the lexical features of the current word to nodes in the dependency structure that have been predicted but not yet lexically instantiated.

This strategy implements two main procedures: ATTACH AS HEAD and ATTACH AS DEPENDENT, respectively. Thus, for each word  $w$ , it first tries to attach  $w$  as the head of words read earlier and then to attach  $w$  as the dependent of a previous word. The main control procedure of the parsing algorithm is thus the same as in Covington (2001). Like the strategy in Covington (2001), processing is immediate, such that parsing proceeds by accepting words and attaching each word into the dependency structure as soon as there is a previous head or dependent it can attach to. In addition, processing is predictive, such that the parser strives to predict subsequent heads and dependents *top-down* as it successively encounters each word in a sentence. As a result, any intermediate representation of the dependency structure may contain nodes which have been predicted but still lack lexical realization. Given that dependency structures lack phrasal nodes, a predicted node only encodes a syntactic function and a lexical category (i.e., a pre-terminal category in constituency terms) which can realize the associated function. Syntactic predictions are derived from the lexical constraints of each successive word encountered during parsing. These predictions represent lexical heads or dependents that will need to be integrated into the dependency structure at some point in the parse. When all words in the input string have been processed, each prediction must be resolved, which means that there must be no lexically uninstantiated nodes left in the dependency structure.

To enable parsing, at least two list-based data structures are required: a buffer, referred to as the *WM* (*working memory*) buffer, where temporary representations of predicted nodes are retained, and a representation of the dependency structure

built thus far, referred to as the *DS* (*dependency structure*). Syntactic predictions are retained in the *WM* buffer until they can be lexically instantiated and attached into the dependency structure. In order to attach the current word  $w$  as head or dependent of a previous word, the lexical features of the current word  $w$  are first matched against a syntactic prediction  $P$  that is part of the *WM* buffer. Such a match results in a retrieval of the previously processed word which fired the prediction of  $P$ . Then, the current word attaches into the dependency structure as head or dependent of this word. Nodes in the *WM* buffer are encoded as either *predicted head nodes* (*PHNs*) or *predicted dependent nodes* (*PDNs*). Each such node is a feature structure with feature values for a syntactic relation and a lexical category which can realize the syntactic relation. In addition, each such node has a feature, *id*, whose value is an integer which identifies the linear position in the input string for the lexical item which fired the current prediction.

### 3.2.3 ATTACH AS HEAD

As each word is accepted from the input its associated lexical features, as defined by  $L_w$ , are looked up in the lexicon. The lexical category of the current word,  $C_w$ , is then matched against the syntactic predictions that are part of the *WM* buffer, such that each successive occurrence of a predicted *head* node in the buffer that matches  $C_w$  is dropped from the buffer and lexically instantiated with the current word,  $w$ . The current word then attaches in the dependency structure as the head of the lexical item identified by the value for *id* in the prediction. Next, the parser attempts to derive top-down predictions about forthcoming dependents of  $w$  based on the lexical constraints associated with the word. If the current word requires a complement to its right, then this prediction is shifted to the buffer. In addition, an incomplete dependency edge is built, in which the current word is attached as the head of a dependent node which is not yet lexically realized.

Consider this procedure with respect to the processing of the word *chased* in the sentence *the black cat chased a rat yesterday*. After having processed *cat*, there is one node in the buffer, corresponding to a prediction of a verb as the head for the subject *cat*. When the word *chased* occurs, this prediction is realized and thus dropped from the buffer and *chased* is attached as the head of *cat* in the dependency structure. Furthermore, the lexical constraints associated with the verb *chased* licenses a new prediction because the verb *chased* requires a complement. Thus, a new expectation that corresponds to an object and a noun for realizing the object function is shifted to the buffer. Furthermore, an incomplete edge is built in the dependency structure, in which *chased* attaches as the lexical head of a dependent node not yet lexically instantiated. Next, control is passed to the procedure which attempts to find a single head for the word being processed.

### 3.2.4 ATTACH AS DEPENDENT

This procedure attempts to attach the current word as a dependent of one of the words previously processed. Given the restriction that every lexical node in a dependency structure is dependent on at most one other lexical node, this procedure is complete as soon as a potential head for the current word is found. The lexical features of the current word,  $w$ , is first matched against the predictions that are part of the buffer, such that the first occurrence of a predicted *dependent* node in the buffer that matches

$C_w$  is dropped from the buffer and lexically instantiated with the current word. This word then attaches in the dependency structure as the dependent of the lexical item identified by the value for  $id$  in the prediction. However, if there is no prediction that matches  $C_w$ , it may only imply that the current word is a word which has not been predicted, e.g. a modifier. Thus it is still possible that the head of  $w$  is part of the previously processed input. In this case, a search process is initiated, beginning with the most recent item  $w_{i-1}$ , moving backwards through previously constructed dependencies. Assuming that the parser only builds projective structures, the parser utilizes the search strategy defined by Covington (2001). Thus, search proceeds by first considering the previous word as a head, then that word's head, its head etc, until either a potential head is found or a node which does not depend on any other is found. If a head is found, then the current word  $w$  is attached into the dependency structure as a dependent of this word. This procedure is then complete and the next word is read from the input. If, however, no head is found in the search process, the parser attempts to predict a forthcoming head node for the current word. Because we are predicting a *head* node for the current word, only left-directed dependencies should be considered. If such a prediction is licensed by grammar, this prediction is shifted to the buffer, causing an incomplete edge to be built in the dependency structure in which the current word attaches as the dependent of a head node not yet lexically realized. However, if the grammar does not sanction such a prediction, the next word is read from the input.

Consider this procedure with respect to the processing of the word *black* in the sentence *the black cat chased a rat yesterday*. When *black* is accepted, there is only one node in the buffer corresponding to a prediction of a head noun for the determiner *the*. Furthermore, search can not be initiated since no complete dependencies have been established thus far in the dependency structure. Thus, the parser attempts to predict an upcoming head for the adjective *black*. Such a prediction is licensed by the grammar, given the rule that adjectives are attributive premodifiers of nouns. Consequently this prediction is shifted to the buffer and an incomplete dependency edge is built in which *black* attaches as dependent of a head node which has yet to be lexically realized. When the following word *cat* is processed, this prediction is realized and *cat* is instantiated as the head of *black* (and also as the head of the determiner *the*) in the structure.

Now, consider this procedure with respect to the processing of the adverbial modifier *yesterday* in the same sentence. After having processed *rat*, the buffer is empty since all predictions associated with the words in the sentence have been resolved and the input string *the black cat chased a rat* constitutes a complete grammatical sentence. When the modifier *yesterday* occurs, a most-recent-first search for its head begins at the previous word *rat*. Since adverbs do not modify nouns, search continues for the head of *rat*. The head of *rat* has been established as *chased* in the structure, and because adverbs do modify verbs, *yesterday* is attached as the dependent of *chased*.

### 3.2.5 Parsing algorithm

Given a list of words to be parsed, and two working lists  $M$  and  $DS$ :

(Initialize)

$M := []$ ; (List of syntactic predictions)

$DS := []$ ; (List of dependency edges)

**while** there are words remaining in the input list:

(Accept word from input)

$W :=$  the word to be processed;

$C_w :=$  the lexical category of  $W$ ;

(ATTACH AS HEAD)

**for**  $PHN :=$  each successive predicted head node in  $M$ ,

**if** [ $funct: Funct, head: C_w, id: D_i$ ] **then**

remove  $PHN$  from  $M$ ;

attach  $W$  as the head of  $D_i$  in  $DS$

**else**

terminate **for** loop

(PREDICT DEPENDENT)

**if** ( $C_w, [funct: Funct, cat: C_d], Mod, right$ )  $\in R$  **then**

$PDN := [funct: Funct, dep: C_d, id: W_i]$ ;

add  $PDN$  to  $M$ ;

add incomplete edge to  $DS$ : ( $W, C_d, Funct, W_i$ )

**end if**

(ATTACH AS DEPENDENT)

**for**  $PDN :=$  each successive predicted dependent node in  $M$ ,

**if** [ $funct: Funct, dep: C_w, id: H_i$ ] **then**

remove  $PDN$  from  $M$ ;

attach  $W$  as dependent of  $H_i$ ;

terminate **for** loop

**else** (Search for the head of  $W$  in  $DS$ )

$H :=$  the word immediately preceding  $W$  in the input string;

$C_h :=$  the lexical category of  $H$ ;

**loop**

**if** ( $C_h, Comp, [funct: F, cat: C_w], right$ )  $\in R$  **then**

attach  $W$  as dependent of  $H$  in  $DS$ ;

terminate the loop

**if**  $H$  is independent **then**

terminate the loop;

**else**  $H :=$  the head of  $H$

**end loop**;

(PREDICT HEAD)

**else if**

( $C_h, Comp, [funct: Funct, cat: C_w], left$ )  $\in R$

or

( $C_h, [funct: Funct, cat: C_w], Mod, left$ )  $\in R$

**then**

$PHN := [funct: Funct, head: C_h, id: W_i]$ ;

add  $PHN$  to  $M$ ;

add incomplete edge to  $DS$ : ( $C_h, W, Funct, W_i$ );

**end while**

### 3.2.6 Parsing illustration

Figure 3.2 illustrates the word-by-word parse sequence for the sentence *the black cat chased the rat yesterday*. The figure illustrates the partial syntactic dependency structure built after each respective word has been input and processed. As shown, parsing proceeds incrementally, i.e. each new word is integrated into the evolving dependency representation as soon as it occurs in the input stream. Furthermore, the parser attempts to establish as much structure as possible at each parse state by predicting, top-down, expected syntactic relations and lexical categories which can realize the associated syntactic relation. In the figure below, we use a bar over nodes that are predicted but not yet found.

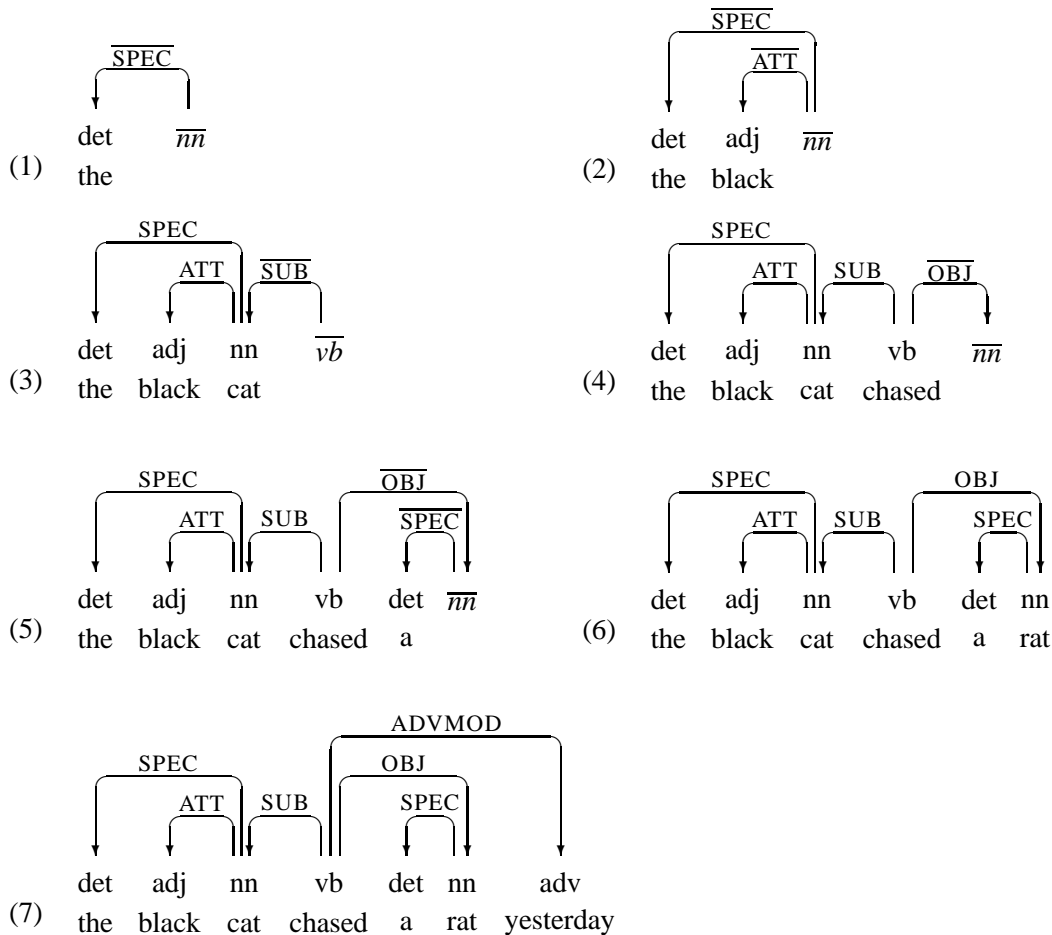


Figure 3.2: Illustration of the word-by-word parse sequence for an English sentence

### 3.3 Storage profiles

The assumption that working memory load increases for each syntactic prediction associated with an incomplete dependency relation affords a simple explanation for many processing contrasts in human sentence parsing. In this section we show that (i) the parsing algorithm presented above is asymmetrical with respect to right-branching

and center-embedded structures, and that (ii) the parsing algorithm associates a higher storage cost with object-relative clause structures than subject-relative clause structures. Given that center-embedded structures require more storage resources than right-branching structures, it follows that object-relative clause structures require more resources than subject-relative clause structures. However, since both pairs of syntactic structures are frequently discussed in the literature, we provide storage profiles for each of the respective structures.

A few things should be noted about the structural dependency analysis assumed here. Syntactic dependencies take precedence over semantic dependencies (Mel’cuk, 1988). With respect to relative clauses this means that a relative pronoun depends on the main verb of the relative clause rather than on the subject of the main clause. The embedded verb is the head of the relative clause, which in turn depends on the nominal element it modifies. Sentence processing, however, also involves linking pronouns to their appropriate antecedents during the incremental structuring of the input. Thus, there is also a semantic dependency relation between a relative pronoun and its antecedent not taken into account here. Furthermore, we do not assume that there are any empty category positions to be associated with extracted relative pronouns in the syntactic representations. Instead, we follow Pickering and Barry (1991) and assume a processing account according to which there is a *direct association* between an extracted pronominal element and its verbal head<sup>6</sup>.

### 3.3.1 Right-branching structures

**Table 3.1:** Word-by-word predictions of storage cost for a right-branching structure

Syntactic Prediction	Input word										
	The	senator	met	the	man	who	attacked	the	reporter	who	slept
Subject		1									
Object			1		R						
Spec	1	R		1	R			1	R		
RC <sub>1</sub> verb						1	R				
RC <sub>1</sub> object							1		R		
RC <sub>2</sub> verb										1	R
Total cost	1	1	1	2	0	1	1	2	0	1	0

Consider the storage cost profile for the right-branching sentence in table 3.1. The storage cost upon processing the first word *the* is 1, corresponding to the prediction of a head noun for the determiner. This prediction is realized when the next word *senator* is processed. At this parse state a new prediction is built, corresponding to a verb for the subject *senator*. Thus, the storage cost is 1 also at the second word. Next, the verb *met* is processed and attached as head for the subject *senator*. Since the verb *met* requires a complement, a new prediction is stored at this point. Processing the next word *the* results in an additional prediction corresponding to a head noun for the determiner. Thus, there is a storage cost of 2 at this point. At the following noun, both predictions are resolved and storage cost is reduced to 0. When the first relative pronoun *who* occurs, a new prediction corresponding to a verb is stored. The next word

<sup>6</sup>See Pickering and Barry (1991) for a sentence processing account which does not rely on empty categories. See also Pickering and Barry (1991) for initial evidence that empty category positions do not have any psychologically real status.

**Table 3.2:** Word-by-word predictions of storage cost for a double center-embedded structure

Syntactic Prediction	Input word										
	The	reporter	who	the	man	who	the	senator	met	attacked	slept
Subject		1									R
Spec	1	R		1	R		1	R			
RC <sub>1</sub> verb			1							R	
RC <sub>1</sub> subject				1						R	
RC <sub>2</sub> verb						1			R		
RC <sub>2</sub> subject							1		R		
Total cost	1	1	2	3	3	4	5	5	3	1	0

*attacked* realizes this prediction. However, the verb *attacked* requires a complement. When the following two words, *the* and *reporter* have been processed storage cost is reduced to 0 again. The following pronoun requires a verb. This prediction is realized upon processing the last word *slept*. The total storage load remains very low during the course of processing the right-branching structure. There are never more than two open predictions at any point in processing the structure.

### 3.3.2 Double center-embedding

The corresponding center-embedded structure of the right-branching structure above is presented in table 3.2. As this table illustrates, there is a considerably higher storage cost associated with processing the center-embedded structure. In particular, after processing each noun and relative pronoun there are corresponding predictions for (i) a verb for a subject and (ii) a verb for the relative pronoun. This results in an increase of verbal predictions which are not beginning to be resolved until the first verb *met* is encountered. The maximal point of storage cost occurs at the most embedded subject *senator* and its preceding specifier *the*. After having processed the word *senator*, there are five verbal predictions corresponding to three verbs for each preceding subject, and two verbs for each of the preceding relative pronouns.

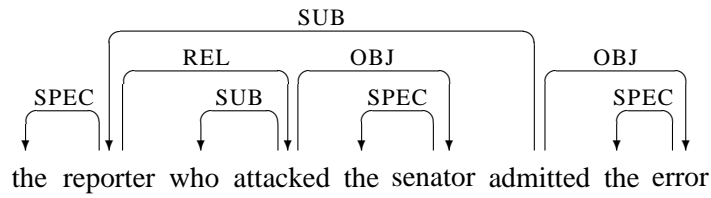
### 3.3.3 Subject- and object-relative clauses

The syntactic dependency representations for the subject- and object-relative clause sentences in (26) and (27) are presented in figure 3.3 and 3.4, respectively.

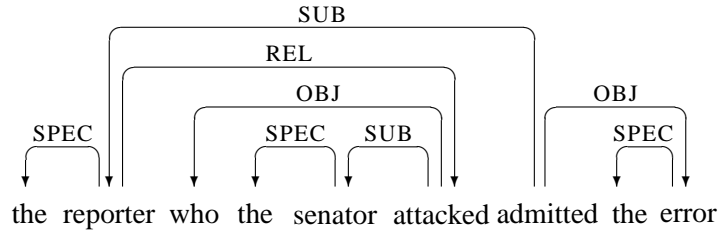
(26) The reporter who attacked the senator admitted the error

(27) The reporter who the senator attacked admitted the error

The storage cost profile for the subject relative clause is presented in table 3.3. At the point of processing the first relative pronoun, *who*, storage cost is incremented to 2 because here not only must the parser store the prediction of a main verb for the subject *reporter*, but also the prediction of a verb for the relative clause. When the verb *attacked* occurs next, the most recent of these predictions are realized. However, at this point an additional prediction corresponding to an object for the embedded verb *attacked* is stored, resulting in a total storage cost of 2. When the determiner *the* occurs next, the parser stores an additional prediction of a head noun for the determiner. When the word *senator* is processed next, it resolves the predictions of a



**Figure 3.3:** Dependency graph for the subject RC sentence in (26)



**Figure 3.4:** Dependency graph for the object RC sentence in (27)

**Table 3.3:** Word-by-word predictions of storage cost for the subject RC structure in (26)

Syntactic Prediction	Input word									
	The	reporter	who	attacked	the	senator	admitted	the	error	
Subject (verb)		1					R			
Object (noun)							1		R	
Det (noun)	1	R			1	R		1	R	
RC verb			1	R						
RC object (noun)				1		R				
Total cost	1	1	2	2	3	1	1	2	0	

complement for the embedded verb and a head noun for the specifier. Thus, storage cost is reduced to 1 at *senator* since there is only one prediction left at this point. The remaining prediction corresponds to a main verb for the subject *reporter*. This prediction is realized upon processing the next word, *admitted*. The verb *admitted* requires an object. This prediction is resolved at the point of processing the last word.

The predictions associated with processing the object relative clause in (27) are presented in table 3.4. The computations differ from the subject relative clause only with respect to the relative clause. When the determiner *the* of the relative clause is processed, the parser makes a prediction corresponding to a head noun for this word. In addition, the parser already maintains two predictions corresponding to a verb for the relative pronoun and a main verb for the subject. Thus, there is a storage cost of 3 at the point of processing this word. When the next word *senator* occurs, the most recently stored prediction is resolved. However, a new prediction corresponding to a verb is now stored. The storage cost thus remains at 3 here.

There is a relatively low storage cost associated with processing both subject- and object-relative clause sentences. However, processing the subject relative clause in (26) requires maintaining one less incomplete prediction than the processing of the object-relative clause sentence in (27).

**Table 3.4:** Word-by-word predictions of storage cost for the object RC structure in (27)

Syntactic Prediction	Input word									
	The	reporter	who	the	senator	attacked	admitted	the	error	
Subject (verb)		1								R
Object (noun)							1			R
Det (noun)	1	R		1	R			1		R
RC verb			1			R				
RC subject (verb)					1	R				
Total cost	1	1	2	3	3	1	1	2		0

## 4 Working memory retrieval

Thus far, we have only considered working memory load as a function of the number of syntactic predictions retained at each word in the processing of a sentence. We have not associated any processing effort with performing structural integrations, i.e. we have not associated any working memory resources with what the dependency locality theory refers to as structural integration cost. To perform an integration in the current parsing model, it is generally necessary to first match the category of the current word with a syntactic prediction that is part of the predictions held in working memory. This match then reactivates the lexical head/dependent associated with the syntactic prediction. In this chapter we turn to consider how structural integration cost is realized in the current model of sentence parsing.

### 4.1 Discourse-based activation decay

According to the dependency locality theory, processing resources are required not only for maintaining temporary predictions associated with incomplete dependencies, but also for integrating new words into the structure built thus far. As noted previously, reading times are assumed to be reflected most directly by the structural integration processes involved in sentence parsing. The DLT structural integration cost is a *distance-based cost*. The greater the distance between the current word  $w$  and a previous head or dependent to which  $w$  attaches in the structure, the greater the integration cost. The reading time at a particular word  $w$  is assumed to increase with the linear distance between  $w$  and an associated head or dependent previously processed. In particular, the difficulty of integrating a new word  $w_2$  as head or dependent of a previous word  $w_1$  is proportional to the number of discourse referents introduced since  $w_1$  was last processed. The total integration cost at a word is the sum of all structural integrations taking place at that word. Distance effects in language comprehension, according to Gibson (2000), could be a reflection of general cognitive constraints on serialized information processing. In particular, distance effects in sentence parsing might be explained in terms of *activation decay* (Gibson, 2000).

In an activation-driven model of sentence comprehension, each representation of a lexical element in working memory has an associated activation level that decays over time as additional words are input and integrated into the syntactic structure. The activation level of a word at a particular point in the processing of a sentence determines its accessibility for retrieval. Integrating a new lexical item as head or dependent of a previous item necessitates a working memory retrieval of some representation of the item which occurred first in the linear order of the dependency relation. In an activation-driven framework, retrieving a previously processed item requires reactivating that item to a target threshold of activation. Since less recent

words will generally be less highly activated than more recent words because of decay, less recent words will be more difficult to reactivate and thus harder to retrieve for additional attachments in the syntactic structure. As noted before, the dependency locality theory assumes that considerable cognitive effort is involved in building a representation for a new discourse referent. Because of the limited quantity of activation in the system, it is assumed that expending this effort causes substantial decays in the activations associated with preceding lexical items (Gibson, 1998).

The structural integration cost metric, however, is a general characterization of the demands different sentence structures place on memory resources, defined without reference to activation decay of working memory representations. Hence, the structural integration cost metric is defined without reference to the more general memory process which is potentially the primary cause for distance effects in sentence parsing. In the computational model presented here, we provide an exploratory account of structural integration cost which instead relies directly on discourse-based activation decay in accounting for distance effects in human sentence parsing. The simplified assumptions we make with respect to discourse-based activation decay is as follows. We assume that the activation level of a word  $w_i$ , at the point of processing any subsequent word  $w_{i+1} \dots w_n$ , is in part a function of the number of discourse referents that have been introduced since  $w_i$  was processed, such that for each discourse referent following  $w_i$  in the input string, there is an associated decrement of activation at word  $w_i$ . The motivation for this assumption is, again, that building a structure for a new discourse referent  $w_i$ , requires the allocation of substantial memory resources, resulting in decreased activations for the lexical items preceding the current word  $w_i$ . Furthermore, we assume that the lexical activation of a word  $w$  decays as new words are introduced and integrated into the dependency structure, *unless* the new words are also involved in a head-dependent relation with  $w$ . If some intermediate word is also involved in a head-dependent relation with  $w$ , then the lexical activation of  $w$  will be reactivated to its target threshold, so that a subsequent retrieval of  $w$  will be less difficult than if the intermediate word were not involved in a head-dependent relation with  $w$ . The difficulty of retrieving a previous head or dependent is determined by the amount of processing resources needed to reactivate the item-to-be-retrieved from its current level of activation to the target level of activation. In the current model, this processing effort is simply a reflection of the number of discourse referents that have been processed since the item was last highly activated. As a new discourse referent is introduced in the input, it is given an initial level of activation set to 1.0, which also serves as the target threshold of activation in the current model. In addition, the introduction of a new discourse referent results in a corresponding decrement of activation set to 0.1 at each preceding verb, nominal and pronominal element. For simplicity, we will initially associate activation decay only with verbs, arguments of verbs and pronominal elements. Thus, activation decay will not be reflected at any other words in the model.

Now, construed in this way, we can consider the discourse-based activation decay process as a lower level realization of the DLT structural integration cost metric. In subsequent sections we shall see how the above assumptions apply in the computational model so as to provide predictions of processing difficulty in accordance with experimental data.

**Table 4.1:** Word-by-word predictions of retrieval cost for the subject RC structure in (28)

Reactivation									
							0.1		error
		0.3						admitted	the 0.9
			0.1			senator	0.9		0.8
				the					
			0.1	attacked		0.9	0.8		0.7
	0.0	who	0.9			0.8	0.7		0.6
		reporter	1.0	0.9		0.8	0.7		0.6
The									
									Decay
Retrieval cost	0.0	0.0	0.0	0.1	0.0	0.1	0.3	0.0	0.1

## 4.2 Retrieval profiles

In this section we show how discourse-based activation decay can account for a number of sentence processing contrasts. Our first example concerns the performance asymmetry between subject- and object-relative clauses. In this first example, we also establish the retrieval cost notation to be used in all examples in this chapter.

### 4.2.1 Subject- and object-relative clauses

Consider the retrieval cost profile for the subject RC sentence in (28), presented in table 4.1.

(28) The reporter who attacked the senator admitted the error

The numbers below the words in the sentence illustrate the activation decay process for each word (except determiners) in a given row read from left to right. Here, we read off the level of activation of some word at the point of processing any subsequent word. For example, the current level of activation of the word *reporter* at the point of processing the word *admitted* is 0.7. This assumption is accounted for as follows. The first discourse referent following *reporter* in the input string is *attacked*. Thus there is a corresponding decrement of activation at *reporter* at the point of processing *attacked*. The next discourse referent following *attacked* is *senator*. Hence, there is an additional decrement of activation at *reporter* also at this point. Next, the word *admitted* occurs. Since verbs are also discourse referents according to the DLT, processing this word causes an additional decrement of activation at *reporter*. Now, given that the initial level of activation for a word is set to 1.0 and that each decrement of activation is set to 0.1, the activation level of *reporter* at the point of processing *admitted* is 0.7. The numbers above the words in the sentence reveal the amount of activation needed to reactivate the word in that *column* from its current level of activation to the target threshold, at the words (*row*) where the parser establishes dependency relations. For example, at the point of processing the word *admitted*, the parser must establish a dependency relation between the verb and the subject *reporter*. Thus, processing the word *admitted* requires retrieving the previous word *reporter*. The retrieval cost associated with reactivating *reporter* is easily read off from the table. The activation level of *reporter* at the point of processing *admitted*

**Table 4.2:** Word-by-word predictions of retrieval cost for the object RC structure in (29)

	Reactivation								
							0.1		error
		0.3						the	0.9
			0.2		0.1	attacked	0.9	admitted	0.8
					senator	0.9	0.8		0.7
				the					
		0.0	who		0.9	0.8	0.7		0.6
		reporter	1.0		0.9	0.8	0.7		0.6
The									
									Decay
Retrieval cost	0.0	0.0	0.0	0.0	0.0	0.3	0.3	0.0	0.1

is established as 0.7. Thus, the retrieval cost for reactivating *reporter* to the target threshold is 0.3, given that the target threshold level is set to 1.0. The retrieval costs at each word in the sentence are presented in the last row of the table. We assume that the total retrieval cost at any word is the sum of all retrievals that take place at that word. The maximal retrieval cost in processing the subject relative clause sentence occurs at the processing of the main verb *admitted*. The total retrieval cost at this word is 0.3.

Now consider the retrieval cost profile for the object RC sentence in (29), presented in table 4.2.

(29) The reporter who the senator attacked admitted the error

There are no retrieval costs associated with processing the first five words. This does not imply, however, that there is no processing effort involved in parsing this region of the sentence. Instead, it is a consequence of the fact that the computational model has separated the processing resources needed for retaining incomplete dependencies in working memory, from the processing resources needed for retrieval of previously processed items. For example, at the point of processing the fourth word, *the*, the parser maintains three predictions. These predictions correspond to a verb for the subject *reporter*, a verb for the relative pronoun *who* and a head noun for the determiner *the*. Now, however, we focus on the retrieval resources needed for integration of new words into the dependency structure built thus far. At the point of processing the embedded verb *attacked*, both the previous relative pronoun *who* and the most recent word *senator* must be retrieved and attached into the dependency representation. At the point of processing the embedded verb *attacked*, the activation level of *who* is 0.8. Thus, there is a cost of 0.2 for reactivating *who* to the target threshold. The current level of activation of the word *senator* when processing the embedded verb *attacked* is 0.9. Thus, retrieving *senator* is associated with a cost of 0.1. Adding both retrievals taking place at the embedded verb *attacked* gives a total retrieval cost of 0.3. at this word. The retrieval cost associated with integrating the main verb *admitted* into the dependency structure is the same as for the subject relative clause, i.e. 0.3.

In summary, the discourse-based decay model detailed here makes the following predictions with respect to actual reading times for subject- and object-relative clause structures. Reading times for the subject relative clause sentence are fast and

**Table 4.3:** Word-by-word predictions of retrieval cost for the right-branching structure in (30)

Reactivation										
									0.1	died
								0.0	who	0.9
					0.1			reporter	1.0	0.9
							the			
				0.1	attacked			0.9		0.8
				0.0	who	0.9		0.8		0.7
	0.1		senator	1.0	0.9			0.8		0.7
		the								
	0.1	met		0.9		0.8		0.7		0.6
	John	0.9		0.8		0.7		0.6		0.5
										Decay
Retrieval cost	0.0	0.1	0.0	0.1	0.0	0.1	0.0	0.1	0.0	0.1

do not increase much except at the main verb *reporter*. In contrast, reading times for the object relative clause are relatively fast for the first five words, then slow both on the embedded verb *attacked* and the main verb *admitted* and then fast again on the last word. These predictions fit the available on-line experimental data for the processing of both kinds of structures (see e.g. Gibson (1998), Gibson (2000), Just and Carpenter (1992)). In addition, Nilsson (2005) compared DLT structural integration costs to actual reading times in a self-paced reading study of Swedish relative clause processing. The results of this study are also consistent with the predictions above, thus lending additional cross-linguistic support for the discourse-based activation decay model presented here.

#### 4.2.2 Right-branching structures

Any computational model that attempts to account for reading times in syntactically complex structures must also account for the processing efficiency and acceptability of simple sentence structures such as deep right-branching structures. It is crucial that the model does not predict any dramatic increase in reading times for such structures. We confirm that there are no such increases predicted for right-branching structures in the current model. The retrieval cost profile for the right-branching sentence in (30) is presented in table 4.3 above.

(30) John met the senator who attacked the reporter who died

The acceptability of right-branching structures such as (30) is accounted for because the retrieval costs remain very low at each word in the processing of the sentence. Retrieval costs do not exceed 0.1. at any parse state. Thus, the current activation-driven parsing model provides the following explanation for the acceptability of right branching structures. Building a structural representation for a right-branching structure requires reactivating lexical items whose level of activation has decayed minimally by the time they are retrieved and integrated into the dependency representation.

**Table 4.4:** Word-by-word predictions of retrieval cost for the double center-embedded structure in (31)

	Reactivation										
		0.5									died
			0.4								0.9
				0.3							0.8
					0.2	0.1	met	0.9	0.8	0.7	0.6
						John	0.9	0.8	0.7	0.6	0.5
					0.0	who	0.9	0.8	0.7	0.6	0.5
						senator	0.9	0.8	0.7	0.6	0.5
				the							
		0.0	who		0.9		0.8	0.7	0.6	0.5	
		reporter			0.9		0.8	0.7	0.6	0.5	
The											
											Decay
Retrieval cost	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.7	0.5	

### 4.2.3 Double center-embedding

We now consider the severe processing difficulty of double center-embedded relative clause structures. The retrieval cost profile for the double center embedded sentence in (31) is presented in table 4.4.

(31) # The reporter who the senator who John met attacked died

The profile reveals that the retrieval costs at the verbs are exceptionally high. In particular, the total retrieval cost at the verb *attacked* is 0.7. Thus, it is extremely difficult to perform the necessary structural integrations at this word. The associated working memory retrievals taking place at this word are as follows. The noun *senator* must be established as the subject of the verb *attacked*. The activation level of *senator* at the point of processing *attacked* is 0.7. Thus, reactivating *senator* is associated with a cost of 0.3. In addition, the relative pronoun *who* of the first embedded clause must also be retrieved and integrated into the dependency structure with *attacked*. The lexical activation of the word *who* at the point of processing the verb *attacked* is 0.6. Thus, there is an additional retrieval cost at the verb of 0.1. Adding both retrievals at *attacked* results in a total retrieval cost of 0.7.

In short, the model predicts the unacceptability of these structures. The general explanation provided here is as follows. Building a complete structural representation for a double center embedded structure requires reactivating lexical items whose activation level has decayed substantially by the time they must be retrieved and integrated into the dependency structure.

### 4.2.4 Embedded pronominal subject in double center embedding

One interesting processing contrast with respect to double center embedded structures is the relative lack of complexity in sentences such as (32) and (33) compared to sentences such as (34).

(32) A book that some Italian that *I* have never heard of wrote will be published soon by MIT Press. (Frank, 1992)

**Table 4.5:** Word-by-word predictions of retrieval cost for the double center-embedded structure in (35)

	Reactivation										
		0.4									died
			0.3		0.2						attacked
						0.1	0.1	met	0.9	0.9	0.9
								I	0.9	0.8	0.8
					0.0	who			0.9	0.8	0.7
					the	senator	1.0		0.9	0.8	0.7
		0.0	who						0.8	0.7	0.6
		reporter	1.0		0.9				0.8	0.7	0.6
The											
											Decay
Retrieval cost	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.5		0.4

(33) The reporter who the senator who *you* met attacked died.

(34) # The reporter who the senator who *John* met attacked died.

When the most embedded subject of a nested relative clause structure is a pronoun, the structure is somewhat easier to process. A study by Warren and Gibson (2000) confirmed that double center embedded structures containing a first- or second-person pronoun as the most embedded subject are perceived as significantly easier to comprehend than the same structures containing a proper name such as *John*, or a definite description such as *the professor*, as the most embedded subject. It should be noted that this contrast can not not explained by the number of incomplete dependencies or predictions retained in working memory. The sentences in (33) and (34) have the same syntactic dependency representation and thus contain the same number of incomplete dependencies at any point. Yet, the sentence in (33) is perceived as easier to process.

This processing contrast, however, is accounted for by the DLT structural integration cost metric and the discourse-based activation decay metric detailed here. Since first- and second-person pronouns, according to the DLT, denote referents which are already implicitly present in the current discourse, less resources are required in order to perform structural integrations across such pronouns.

Consider the retrieval cost profile for the double center embedded sentence in (35) presented in table 4.5.

(35) The reporter who the senator who I met attacked died

During the course of processing this sentence it is necessary to perform the same working memory retrievals as in processing the previous center embedded sentence in (31). For instance, when the verb *attacked* occurs, both the noun *senator* and the pronoun *who* of the first embedded relative clause must be reactivated to the target threshold level. However, these memory retrievals are less difficult in the center embedded structure which contains a pronoun as the most embedded subject, because the words *senator* and *who* have decayed less than at the same point in processing (31). The maximal retrieval cost in (35) is 0.5 and occurs at the verb *attacked*. The maximal retrieval cost in (31) occurs at the same word but instead at a retrieval cost

**Table 4.6:** Word-by-word predictions of retrieval cost for the sentence in (36). Local attachment reading

Reactivation									
						<b>0.1</b>			yesterday
						0.1		country	0.9
							the		
		0.2		0.1	left			0.9	0.9
					detective	0.9		0.8	0.7
			the						
	0.1	said		0.9	0.8			0.9	0.8
	reporter	0.9		0.8	0.7			0.6	0.5
The									
									Decay
Retrieval cost	0.0	0.0	0.1	0.0	0.0	0.3	0.0	0.1	0.1

of 0.7. Hence, it takes less resources to reactivate the words *senator* and *who* in (35) compared to (31).

In sum, the current model accounts for the processing contrast as follows. Processing a double center embedded structure containing a first- or second- person pronoun as the most embedded subject requires reactivating items whose activation level has decayed to a less extent by the time they must be retrieved, compared to a double center embedded structure which does not contain any first- or second-person pronouns. Thus center embedded structures with an embedded pronominal subject are perceived as easier to comprehend.

#### 4.2.5 Recency effects in structural ambiguity resolution

Distance effects are reflected not only in the processing of unambiguous structures, but also in the processing of ambiguous structures. In this section we turn to consider the predictions that the discourse-based activation decay model makes when used as part of a metric in structural ambiguity resolution.

All other things being equal, the human sentence processing system prefers to attach an ambiguous modifier to the most recent possible site in the structure (e.g. Frazier 1978). Thus, for example the sentence in (36) below is associated with a strongly preferred local reading.

(36) The bartender said the detective left the country yesterday.

In (36), the adverbial modifier *yesterday* can attach either to the most local verb *left* or to the more distant verb *said*. There is a strong preference, however, to attach *yesterday* to the most recent verb *left*. According to the dependency locality theory, such locality preferences occur because they keep the distribution of working memory resources at a minimum during each stage in processing. Thus, the same mechanisms that determine the processing difficulty of a sentence also influence the resolution of structural ambiguities:

*Ambiguity resolution hypothesis* (Gibson, 2000, 115)

In choosing among ambiguous structures, two of the factors that the

**Table 4.7:** Word-by-word predictions of retrieval cost for the sentence in (36). Non-local attachment reading

Reactivation									
			<b>0.2</b>				<i>0.1</i>	country	yesterday
								the	0.9
			<i>0.2</i>	<i>0.1</i>	left			0.9	0.9
				detective	0.9			0.8	0.7
				the					
		<i>0.1</i>	said	0.9	0.8			0.9	0.8
		reporter	0.9	0.8	0.7			0.6	0.5
The									
									Decay
Retrieval cost	0.0	0.0	0.1	0.0	0.0	0.3	0.0	0.1	0.2

processor uses to evaluate its choices are DLT storage and structural integration cost (in addition to informational constraints, such as lexical frequencies, plausibility, and context.).

Locality preferences in ambiguous structures are accounted for in the DLT under the assumption that the human sentence processing system follows the less complex structure in terms of structural integration and storage cost. Discourse-based activation decay provides an explanation for the preferred reading in (36). Consider the retrieval cost profiles for each reading of (36), presented in table 4.6 and 4.7. The two profiles differ only with respect to the retrieval cost at the last word *yesterday*. In the local attachment reading where *yesterday* modifies *left*, the cost for retrieving *left* is 0.1. In the non-local attachment reading where *yesterday* modifies *said*, the cost for retrieving *said* is 0.2. In this discourse-based activation decay framework, the local attachment is preferred because it takes less resources to reactivate the verb *left* since it occurred more locally in the input string and its activation level has decayed less than that of the verb *said*.

## 5 Implementation

The computational model of working memory load in sentence processing presented in this thesis has been realized as an implemented system which is freely available for research and educational purposes<sup>1</sup>. The system is designed to recover dependency-based representations of sentence structure and to provide detailed predictions of working memory load at each word as a sentence is incrementally processed. Thus, the general task of the system is to map the linguistic complexity of sentence structure to relative comprehension times in reading. Each sentence is assigned a dependency representation and a profile which details the processing demands associated with building a structure for it. This profile can be compared to empirical data such as reading-time data. The system implements the parsing algorithm in section 3.2.5 above. In addition, each structure building parser operation which necessitates a retrieval of a previous lexical node is associated with a retrieval cost. The retrieval process is implemented in accordance with the discourse-based activation-decay principles reviewed in chapter 4.

A fully functional parser must specify mechanisms for handling local lexical and structural ambiguity. We have not detailed any specific strategy for dealing with ambiguity, in part due to the fact that our focus is on working memory processes in sentences independent of ambiguity. In the present implementation we rely on backtracking in order to resolve local ambiguities.

The system has been implemented in such a way as to facilitate the understanding of the procedures involved. This means that there is a relatively high degree of transparency between the computational model presented here and the actual details of the implementation.

---

<sup>1</sup>URL: <http://stp.lingfil.uu.se/~tias>

## 6 Summary

This work presented a computational model of working memory load in real-time sentence processing that accounts for a number of well-established linguistic performance effects. The model is based upon the assumption that there are working memory demands in on-line comprehension associated with both storage of incomplete syntactic structure and integration of new lexical input into the structure built thus far. This assumption is in accordance with a recent sentence processing theory - the dependency locality theory -, as well as with recent conceptions of working memory load in other cognitive information processing tasks. The model presented here has been realized as an implemented system for sentence parsing which yields detailed predictions of word-by-word processing difficulty and on-line reading times. We restricted ourselves to consider a small set of linguistic structures, for which there are substantial experimental evidence of particular processing effects. We have shown that the model is successful in accounting for the processing effects associated with these structures. In particular, the implementation accounts for the following performance effects:

1. The processing asymmetry between subject- and object- relative clause structures
2. The unacceptability of multiply center-embedded relative clause structures
3. The lower complexity of multiply embedded structures with pronominals in the most embedded subject position
4. The acceptability of right- and left-branching structures
5. Recency effects in ambiguous structures.

In contrast to current sentence processing theories which rely on phrase structure representations, we presented a parsing model where the basic syntactic relations necessary for interpretation are established directly as asymmetrical dependency relations between lexical nodes. Thus, the parser produces dependency-based representations of syntactic structure. These representations can be defined as directed acyclic graphs. We presented a parsing algorithm for parsing a fragment of English into such representations. The algorithm was motivated by a concern to incorporate a predictive component into the parsing procedure. The resulting algorithm is driven by both bottom-up and top-down processing and reflects both the incremental and predictive nature of human sentence processing. A simple metric of working memory load, the number of predicted heads and dependents associated with incomplete dependencies, were shown to account for a number of performance effects. Thus, in accordance with the dependency locality theory, the parsing model predicts that

processing difficulty increases when the number of syntactic predictions retained in working memory increases.

Structural integration cost is of particular relevance to a computational model that aims to make strong testable predictions of processing load, as it is supposed to reflect reading times most directly. The sentence processing model presented here implements a novel formulation of structural integration cost which affords an account of the retrieval processes which give rise to integration costs in sentence processing. This account is defined in terms of activation-decay, a cognitive principle generally assumed to influence any serialized information processing task. The difficulty of retrieving a previously processed head or dependent node  $n_1$ , in order to attach it in the structure with the current lexical node  $n_2$ , depends on the current level of activation of  $n_1$ . The current level of activation of  $n_1$  is a decreasing function of the number of discourse referents which have been processed since it was last highly activated. The more discourse referents which have been introduced in the interim, the lower the activation level of  $n_1$ , and as a result, the higher the cost for retrieving it and attaching it into the structure with the current node  $n_2$ .

In addition to accounting for behavioral data on the linguistic structures considered in this thesis, the implemented model lends an opportunity to further investigate its predictions with respect to other linguistic structures. If these predictions remain consistent with experimental evidence such as reading time data, then the computational model presented here may provide additional support in articulating the ways in which working memory constrains the efficiency of sentence processing. As such, it will have implications for psycholinguistic research as well as for natural language engineering.

# Bibliography

- Abney, S. P. and Johnson, M. (1991). Memory requirements and local ambiguities of parsing strategies. *Journal of Psycholinguistic Research*, 20:233–250.
- Aho, A. V. and Ullman, J. D. (1973). *The theory of Parsing, translation, and compiling*, volume 1: Parsing. Prentice-Hall, Engelwood Cliffs, N.J.
- Caplan, D. and Waters, G. S. (1999). Verbal working memory and sentence comprehension. *Behavioral and Brain Sciences*, 22:77–126.
- Chomsky, N. and Miller, G. A. (1963). Introduction to the formal analysis of natural languages. In Luce, R. D., Bush, R. R., and Galanter, E., editors, *Handbook of Mathematical Psychology*, volume II, pages 269–321. John Wiley, New York.
- Covington, M. A. (2001). A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.
- Cowper, E. A. (1976). *Constraints on sentence complexity: A model for syntactic processing*. Ph.D. dissertation, Brown University, Providence, R.I.
- Ford, M. (1983). A method for obtaining measures of local parsing complexity throughout sentences. *Journal of Verbal Learning and Verbal Behavior*, 22:203–218.
- Frank, R. (1992). *Syntactic locality and tree-adjoining grammar: Grammatical, acquisition, and processing perspectives*. PhD thesis, University of Pennsylvania, Philadelphia.
- Frazier, L. (1987). Theories of sentence processing. In Garfield, J. L., editor, *Modularity in Knowledge Representation and Natural Language Understanding*. MIT Press.
- Gibson, E. (1991). *A computational theory of human linguistic processing: Memory limitations and processing breakdown*. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh.
- Gibson, E. (1998). Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68:1–76.
- Gibson, E. (2000). The dependency locality theory: A distance-based theory of linguistic complexity. In Miyashita, Y., Marantz, A., and O’Neil, W., editors, *Image, Language, Brain: Papers from the first mind articulation project symposium*. MIT Press, Cambridge, MA.

- Grodzinsky, Y. (1989). Agrammatic comprehension of relative clauses. *Brain and Language*, 31:480–499.
- Hudson, R. A. (1984). *Word Grammar*. Blackwell.
- Johnson-Laird, P. N. (1983). *Mental Models: towards a cognitive science of language, inference and consciousness*. Cambridge, UK: Cambridge University Press.
- Just, M. A. and Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, 99:122–149.
- Just, M. A. and Carpenter, P. A. (1980). A theory of reading: From eye fixations to comprehension. *Psychological Review*, 87:329–354.
- Kamide, Y., Altmann, G. T. M., and Haywood, S. (2003b). Prediction and thematic information in incremental sentence processing: Evidence from anticipatory eye movements. *Journal of Memory and Language*, 49:133–156.
- Kamide, Y., Scheepers, C., and Altmann, G. T. M. (2003a). Integration of syntactic and semantic information in predictive processing: Cross-linguistic evidence from German and English. *Journal of Psycholinguistic Research*, 32:37–55.
- King, J. and Just, M. J. (1991). Individual differences in syntactic processing: The role of working memory. *Journal of Memory and Language*, 30:580–602.
- Lewis, R. (1993). *An architecturally-based theory of human sentence comprehension*. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh.
- Lewis, R. (1996). 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. and Vasishth, S. (2005). An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, 29:1–45.
- Marslen-Wilson, W. (1973). Linguistic structure and speech shadowing at very short latencies. *Nature*, 244:522–533.
- Mel'cuk, I. (1988). *Dependency Syntax: Theory and Practice*. State University of New York Press.
- Miller, G. A. and Chomsky, N. (1963). Finitary models of language users. In Luce, R. D., Bush, R. R., and Galanter, E., editors, *Handbook of Mathematical Psychology*, volume II, pages 419–491. John Wiley, New York.
- Nilsson, M. (2005). An implemented account of integration and storage in Swedish sentence processing. In *Proceedings of the 11th Annual Conference on Architectures and Mechanisms for Language Processing*.
- Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In Van Noord, G., editor, *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

- Nivre, J. (2005). Dependency grammar and dependency parsing. Technical Report MSI report 05133., Växjö University: School of Mathematics and Systems Engineering.
- Pickering, M. and Barry, G. (1991). Sentence processing without empty categories. *Language and cognitive Processes*, 6:229–259.
- Pylyshyn, Z. W. (1984). *Computation and Cognition*. Bradford / MIT Press, Cambridge, MA.
- Resnik, P. (1992). Left-corner parsing and psychological plausibility. In *Proceedings, COLING-92*, pages 191–197.
- Roeck, A. D., Johnson, R., King, M., Rosner, M., Sampson, G., and Varile, N. (1982). A myth about center-embedding. *Lingua*, 58:327–340.
- Warren, T. and Gibson, E. (2000). Effects of discourse status on reading times. Poster presented at 13th CUNY conference on sentence processing.