

Representing a Parse in the Brain: The TPARRSE Model

Carol Whitney¹ and Amy Weinberg²

¹Neural and Cognitive Sciences Program
Department of Computer Science
University of Maryland

²Department of Linguistics
University of Maryland

Address correspondence to:

Carol Whitney
Department of Computer Science
University of Maryland
College Park, MD 20742

cwhitney@cs.umd.edu

ABSTRACT

The human parser takes a string of words (a sentence) as input and creates a representation of meaning as output. Physical constraints require that this output be calculated and represented on a neural network having finite resources and fixed connectivity (on the time scale of parsing). Representational and processing constraints require encoding of hierarchical bindings, and access to individual sub-structures. Based on these requirements, a two-track model of sentence processing is proposed. The hierarchical representation of meaning is encoded via combinatory vector operations. Subservicing this processing and allowing access to individual items, a working memory temporally encodes syntactic relationships. A parsing algorithm which operates over these distributed representations is presented, and simulated at the symbolic level. The proposed architecture explains a range of psycholinguistic phenomena associated with the perception of syntactic complexity, providing a more comprehensive account than the Dependency Locality Theory (Gibson, 1998; 2000).

1 Introduction

We seek to construct models of language processing that are constrained from the bottom up by neurobiological plausibility. Any model of language processing must explain how a linear string of words is mapped to a hierarchically encoded meaning representation. Neurobiological considerations make this a very hard problem and impose stringent constraints on this mapping, as we will see below. Human languages contain potentially unbounded recursive structures. Neurobiologically inspired models have typically proposed distributed representations combining large vectors to handle this type of recursion. The distributed nature of this representation however, does not square well with the compositional property of natural language grammars. Sentences are formed by component phrases, and the parser must be able to retrieve well-specified components of structure (as opposed to the whole string) at various points during analysis. In this work, we will propose an architecture that adds a temporal encoding mechanism that allows access to sub-structure encoded as vectors. This addition overcomes an otherwise important limitation of the vector encoding. Surprisingly, we will see that an architecture designed to meet these constraints gives a natural account of a variety of psycholinguistic phenomena dealing with sentence complexity.

In this initial work, we concentrate on the neural representations which support parsing. We investigate how the output of the parser could be encoded, and how it could be generated from a string of words with the help of working memory. The output and working-memory representations will be discussed at the neural level, while parsing rules which operate over these representations will be specified at the algorithmic level.

Almost all models of syntactic processing assume that the output specifies a mapping between the linear string and its thematic interpretation. Accordingly, we assume that output of the parser is a *thematic tree*, a hierarchical representation of the sentence in terms of thematic roles, which indicate “who did what to whom”. Calculation and representation of this thematic tree requires three important operations: binding, embedding, and reaccessing previously processed elements. We first discuss these requirements, and then consider how they could be met within a neural architecture.

In terms of thematic roles, the “who” is called the *Agent*, and the “whom” is called the *Theme*, while a predicate (verb) specifies the “what”. For example, the interpretation of:

1. John loves Alice.

is that *John* is the Agent¹, *Alice* is the Theme, and *loves* is the predicate. Thus, a specification of thematic roles requires association of items with particular roles. This is analogous to the well-known *binding problem* in neuroscience, wherein two separate representations must be unified in real time.

Natural language is inherently hierarchical, allowing phrases and clauses to be embedded within other phrases and clauses. For instance, a noun phrase (NP) which satisfies a thematic role may itself contain thematic-role bindings. Consider the sentence:

2. When the dog that Mary adopted bit her, she screamed.

¹We will use the term “Agent” in a broad sense, and not distinguish between more fine-grained thematic roles such as Agent and Experiencer, etc.

Here the relative clause (RC) *that Mary adopted* specifies a set of thematic roles, and is embedded within an NP which is the Agent of a higher level predicate, *bit*. In principle, we can understand sentences containing unbounded hierarchical structure. Thus, specification of thematic roles requires the ability to represent these embeddings.

It is well known that any processing system must compensate for the fact that many important semantic and syntactic relations occur between elements that are not adjacent (in fact, can be unboundedly far away from each other) in the linear string. For example, the RC in the above sentence interrupts dependencies in the higher clause *the dog bit*. Thus when *bit* is encountered, processing of the RC must be completed, and processing of the adverbial must be resumed, with *the dog* selectively re-accessed as the Agent of *bit*. Thus, data structures must support the computation of the correct thematic relations, ignoring intervening unrelated material and returning to the correct point in the higher clause. These cases (interrupted or non-local dependencies) have traditionally been used to constrain processing models. We will make similar use of these cases here.

On a digital computer, hierarchical structure is represented and accessed by referring to sub-structure by its memory address. This solution is not available within a neural network, since such a system does not support access by memory address. As a result, encoding hierarchical bindings and handling non-local dependencies have posed serious problems for previous neurobiologically inspired models, as we discuss next.

How might these requirements be handled in a neurally plausible architecture, comprised of a finite number of processing units having fixed connectivity? The most straightforward way of performing thematic-role binding is for a certain set of cells represent the Agent, a different set of cells represent the verb, and another set the Theme. That is, each set of cells represents a specific role, and the activity over each set represents the item corresponding to that role. However, this scheme does not allow embeddings, because there is no way to represent that an item is itself composed of other items. Therefore this scheme does not allow encoding of the multiply embedded structures in natural language.

An alternative possibility is that hierarchical structure is represented directly via the connections between cells. For example, the concept *Mary* is represented by a certain set of cells being active, and the concept *Agent* is represented by a different set. The joint firing of these cells activates another set encoding that *Mary* is an *Agent*. Since the connectivity of cells is fixed on the time scale of seconds, it is not possible to dynamically create new units as needed. Therefore cells with the proper connectivity would have to already exist. However, this leads to an exponential explosion in the number of cells that required to represent any possible sentence. While Shastri (2001) has argued that it is plausible for cells connecting any pair of concepts to exist, this analysis neglects the hierarchical nature of language - cells connecting any pair of pairs would also have to exist to handle the full range of syntactic structures. Thus such a representation does not seem viable.

There have been two recent attempts to generate pairings with a fixed connectivity and finite resources. While neither suffices to handle the totality of problems presented by natural languages, we will propose features of the two approaches to do the job.

One possibility is to represent relationships between concepts by the timing of firing of the constituent concepts. Shastri and Ajjanagadde (1993), Shastri (1999), and Hummel and Holyoak (1997) have used synchronous firing to encode thematic binding relations for certain classes of propositions/sentences. For example, *Mary* is encoded as an Agent by having the

concepts *Mary* and *Agent* fire together. Other pairings would fire synchronously during other time slots. However, a weakness of the temporal approach for a parse representation is that time is linear. It is not clear how to map hierarchical structure onto a linear encoding in a general manner. One possibility, adopted by Hummel and Holyoak (1997), is for all of the nodes along some path of the tree to fire synchronously. Each path fires in a different temporal slot, until the whole tree is traversed. However, this raises the issue of how the cells representing the nodes are recruited, and how proper timing is coordinated amongst these cells. In the Hummel and Holyoak model, timing was coordinated by excitatory and inhibitory connections. Thus the desired tree had to already be directly encoded by the connectivity between cells in order to instantiate the temporal encoding. So this representation suffers from the same problems as discussed above. Shastri and Ajjanagadde (1993) assumed that temporal pairings could be generated on the fly. However, they did not address how this was accomplished, nor how nested relationships would be represented. Thus, no one has shown how a temporal encoding of hierarchical structure could be generated on the fly in a general way. This requires solving both problems of how the structure is mapped onto the temporal encoding, and how this temporal encoding is activated during processing.

Another approach to hierarchical encoding is to represent each item by a large vector, and to define operations which combine vectors. For example, *Mary* and *Agent* would each be a vector, and they would be combined to form a new vector representing $Agent = Mary$. This new vector could then be combined with other vectors to produce a hierarchical encoding. To avoid exponential explosion, and to allow calculations to be performed iteratively over a fixed set of cells, the combinatory operations should yield a vector that is the same length as the constituent vectors. Thus the combination is a reduced representation (RR) of the constituent vectors (Hinton, 1990).

Pollack (1990) proposed a scheme wherein the reduced representation is comprised of the hidden units' activations in a network trained by backpropagation to auto-associate. Rohde (2002) used such an approach in a system which developed an RR representation of the syntactic structure of a sentence. This encoding could be queried to yield the relationships specified by the sentence. Such an approach has the advantage that the rules of processing (the grammar) are learned along with the representations. However, we will see below that such a representation is not actually robust enough to encode arbitrary hierarchical structure.

2

A different approach is to define combinatory vector operators with the desired properties. Plate (1995) has proposed a binding scheme for real-valued vectors, based on the convolution of their outer product. Kanerva (1995) has proposed a scheme that operates in a bit-wise fashion over binary vectors. These proposals each employ two different combinatory operators, allowing systematic encoding of arbitrary bindings and embeddings.

While it is more straightforward to encode structure in an RR encoding than in a temporal one, an RR encoding has other problems. By definition, an RR encoding is a distributed one; information about any relationship is distributed across the entire vector. Once an item is incorporated into a distributed encoding, it has lost its identity, and is not directly accessible. However, as previously discussed, the parser must be able to easily re-access

²For example, for novel sentences containing 6 thematic-role bindings, the system was only able to correctly bind all NPs to their appropriate thematic roles approximately 50% of the time.

previously processed items.

Furthermore, natural language is full of ambiguity, which can create the need to reinterpret previously processed material. For example, in the sentence starting:

3. John gave the baby ...

the baby could be the receiver (Goal) or the thing given (Theme). Recent EEG studies suggest that the parser is serial (Hopf, Bader, Meng & Bayer, 2003), keeping a single parse representation which is then modified if the initial interpretation turns out to be incorrect. Thus, if *the baby* is initially interpreted as the Goal, but turns out to be Theme, the *Agent = John* relationship should be maintained, while the *Goal = the baby* relationship should be changed to *Theme = the baby*. However, such a selective change cannot be carried out directly on a distributed representation. Rather, it would require decoding all the relationships, storing them somehow, and recoding them all with the undesired binding replaced by the desired binding. Thus while a distributed representation is well-suited for representing the final thematic tree, it is ill-suited for interim processing which requires selective access to and modification of information.

While an RR encoding does not support easy access to individual items, a temporal encoding does not suffer from this problem. In a temporal encoding, each component item maintains its individuality, and each relationship is segregated into a separate temporal slot. Thus it is easy to access a single item or to directly delete a relationship between two items by inhibiting those items, without affecting other relationships.

In summary, a temporal encoding provides segregation of information, but does not straightforwardly encode hierarchical structure. An RR encoding provides ease of encoding hierarchical structure, but does not provide segregation of information. In the remainder of this article, we propose a parse representation that combines the temporal and RR approaches. This two-track model capitalizes on the strengths of each type of representation: a temporal encoding yields a Working Memory (WM) which supports the generation of an RR encoding of the thematic tree. The temporal encoding is produced in parallel with the RR encoding, and serves as a backup representation if subsequent processing reveals a need to re-access previously processed sub-structure, as in interrupted dependencies or reanalysis. In languages in which verbs follow their objects, the temporal encoding plays a more significant role in generating the RR encoding. We dub this model TPARRSE (Temporal Parsing And Reduced Representation Semantic Encoding).

Our goal in this article is to demonstrate that the proposed processing is powerful enough to handle the range of recursive structures found in English and other languages. We will show then that constraints which are required to make this model neurobiologically plausible impose restrictions that yield complexity metrics that allow us to handle a range of observed psycholinguistic phenomena dealing with structures that are *not* easy to comprehend.

The remainder of the paper is divided into four main sections. In the first of these, we focus on the output representation (RR encoding). That is, we specify the representation of hierarchical structure, which is based on Kanerva (1995). We then present a parsing algorithm for right-branching sentences.

In the next section, we turn to the temporal encoding. We specify a WM representation of syntactic structure, which is used to process long-distance relationships. We enhance

the parsing algorithm to make use of this representation, allowing processing of arbitrarily complex sentences.

In the following section, we present computational demonstrations of the viability of our proposed representations and algorithms. This section presents low-level details of the implementation of the RR and WM encodings within a neural framework, and a high-level simulation of the parsing algorithm. We compare our model to other distributed approaches, such as Rohde (2002).

We then discuss psycholinguistic complexity data, explaining why certain unambiguous sentences seem too hard to understand. We show that problems with synchronizing information both within and across the RR and WM encodings can yield parser breakdown. We conclude this section by contrasting TPARRSE with other accounts of complexity, focusing on the Dependency Locality Theory (Gibson, 1998; 2000).³

2 The RR encoding

The ultimate goal of processing language is to understand “who did what to whom”. We assume that these thematic relationships are mapped onto a distributed encoding that can capture the relations traditionally expressed in a syntactic tree. This section deals with the specification of that mapping. We start by describing how the types of relationships normally encoded in a tree are represented in an RR encoding. The proposed representation is based on the need to unambiguously represent the structures found in natural languages, linguists’ insights about those structures, and the need for incremental processing. After we have specified the RR encoding of the thematic tree, we then consider how it is generated from a string of words. We first address the simplest constructions - unambiguous sentences that are right branching.

2.1 Specification of the RR encoding

In analyzing language, verbs are often thought of as functions, or *predicates*, which take *arguments*. For example, the verb *loves* is a function that takes Agent and Theme arguments, and specifies a relationship between those two entities. The number and type of these argument categories are determined by the predicate. *Love* takes two arguments, while a verb like *sleep* only takes one. Other parts of speech can also impose similar restrictions. For example the adverb *because* requires an entire clause or proposition for syntactic and semantic completeness:

4. *Because John,
Because John is sick,

A category whose occurrence is not restricted by any semantic feature of another category, but rather co-occurs with any member of a part-of-speech class, is called a modifier or adjunct. For example, the time modifier *on Tuesday* may appear with any verb.

³In subsequent work, we show how these mechanisms also account for the relative difficulty of reanalysis cases (Whitney & Weinberg, in preparation).

5. I slept on Tuesday.
I loved the movie on Tuesday.

The presence of a subject is required whenever there is a complete verb phrase, even if the subject adds nothing to the meaning of a sentence as in:

6. It seems that John is smart.

Because the occurrence of a subject is not dependent on the semantic features of the verb, linguists consider subjects to be like adjuncts (Marantz, 1984).

We propose an RR encoding of these relationships based on Kanerva (1995). The basic unit of representation is a large (dimension $\approx 10,000$), random, binary vector. For the purposes of this article, we will represent the component words used to construct a syntactic representation as though they were unitary items, but we assume that these are also composed from some distributed representation whose details we leave for future research.⁴ Items (vectors) will be given in boldface.

Two types of combinatory operators, *merge* and *bind*, combine items into hierarchical structures. These operators each take two vectors and create a resultant vector that is of the same size as the original vectors. Thus, there is no exponential explosion when building composite structures. The merge operator creates a new item which is similar to the two items which comprise it, while the bind operator creates a new item which is not similar to its constituent items. We use the symbol ‘+’ to represent merge, and the symbol ‘@’ to represent bind. There are also inverse operators which decode an item, breaking it down into its constituent items. Under these operators, bind distributes over merge; that is, decoding $\mathbf{a} @ (\mathbf{b} + \mathbf{c})$ gives the same result as decoding $\mathbf{a} @ \mathbf{b} + \mathbf{a} @ \mathbf{c}$.⁵ For the following discussion of representations and algorithms, characterization of these operations at this high level is sufficient. The low-level details are given in section 4.1.

In general terms, we use the bind operator to specify what is normally thought of as a complement relationship in a syntactic tree (i.e., a head and its sister). We use the merge operator to join together other parts of the tree. More specifically, we use bind to specify the predicate-argument relationships, and merge to join arguments and adjuncts.

For example, the RR encoding of :

7. Sue kissed Bill on Saturday.

is:

sue + kissed@bill + When@on@saturday

This encodes that *Sue* is the Agent of *kissed*, and *Bill* is the Theme of *kissed*. *Saturday* is the object of *on*, and the prepositional phrase (PP) *on Saturday* modifies *kissed* by telling

⁴We will not present the details of how the input words and morphemes are composed, and how output semantic relations beyond thematic roles are represented. As a sketch, we assume that each base vector (i.e., uncomposed vector representing a word or morpheme) is associated, via connection weights, with a neural network that encodes meaning via connectivity within that network. That is, the vector does not itself encode semantic information, but rather acts as a “pointer” to a semantic encoding which uses a different neural code. The nature of the semantic code is beyond the scope of this article.

⁵ $\mathbf{a} @ (\mathbf{b} + \mathbf{c})$ is not exactly the same vector as $\mathbf{a} @ \mathbf{b} + \mathbf{a} @ \mathbf{c}$

when it occurred. The verb's arguments, and adjunct (the PP) are joined together by the merge operator.

Verbs and prepositions are bound directly to their objects, as the presence of these categories is predicted by the semantic properties of that verb or preposition. The Agent of the sentence remains unbound. Other semantically determined relationships are represented by special predefined items (identified with capital letters). A verb adjunct fulfills a optional thematic role which is specified by a predefined item (**When**, in this example). A predicate must always take an argument; the object argument of an intransitive verb is a filler item, **E**.

In a passive sentence, the subject is the Theme and not the Agent, as in the sentence:

8. Bill was kissed by Sue on Saturday.

Because this sentence expresses the same thematic relationships as (7), it has the same RR encoding as that sentence.⁶

If an argument is itself a clause, the same rules recursively apply. For example, in:

9. Mary said that the man arrived from the beach.

the Theme of the verb *said* is the sentential complement *the man arrived from the beach*. The RR encoding is:

john + said@(the + man + arrived@E + Where@from@(the + beach))

Here **said** is bound to the encoding of its sentential complement.⁷

Traditional syntactic representation uses geometry to encode semantic dependencies. Potential argumenthood depends on being in a particular geometrical relation, *dominance*, even if there is no direct link between the predicate and its argument. By dividing categories into phrases, a notion of co-constituency is established that is used to constrain syntactic operations. For example, categories can be moved in a phrase by phrase manner, as in:

10. *Into the room* he walked.

We delineate these relations and establish a chain of dominance by defining a notion of *enclosing scope*. This term refers to the categories that an item **x** is bound to. The enclosing scope determines where **x** is attached in the thematic tree.

For example, the sentential complement above is dominated by the verb **said**. Thus, the enclosing scope of all the constituents in the complement clause is this verb. This gives an implicit representation of co-constituency. Because **Where @ from @ (the + beach)** has the same enclosing scope as the verb **arrived**, it modifies that verb. The enclosing scope of the PP **from @ (the + beach)** is **Where**, which specifies a thematic role, indicating that this PP modifies a verb. If it were not bound to a thematic role, the PP would modify the subject of the complement clause instead, yielding a reading equivalent to:

⁶There may be additional information in the RR encoding indicating the voice of the verb. The point is that the thematic-role encoding is invariant.

⁷We only consider sentential clauses whose complementizer does not add additional semantics. That is, we don't consider sentences involving wh-movement, such as *John knows when Mary came*. Such sentences would have to be handled differently, in order to retain the complementizer.

11. John said that the man from the beach arrived.

Since `bind` distributes over `merge`, the RR encoding of the above example is equivalent to:

john + said@(the + man) + said@arrived@E + said@Where@from@(the + beach)

Thus, an RR encoding is comprised of NPs having various enclosing scopes; each NP's enclosing scope specifies its role. For example, the enclosing scope of **the + beach** is **said @ Where @ from**, indicating that this NP is the object of the preposition **from**; this PP specifies the thematic role **Where** and modifies the verb having the enclosing scope **said**. Because of this property, an RR encoding can be constructed by maintaining the enclosing scope that applies to each incoming NP, as we see next.

2.2 Generating the RR encoding

We now turn to how we generate the RR encoding incrementally during parsing. For now, we address unambiguous, right-branching sentences. These structures are easy to process because there are no incomplete dependencies in the parent clause when the embedded clause is introduced. Therefore, the RR encoding of such sentences can be produced directly from the input, without relying on the temporal representation.

2.2.1 Two Stage Architecture

There are two stages within RR encoding. The first stage groups together nouns and adjectives into nominals, prepositions and nominals into prepositional phrases, and verbs and their auxiliaries into verbals. If a prepositional phrase modifies a nominal, it is combined with the nominal into a noun phrase. These categories are the basic building blocks of the predicate argument structures. When a new word signals the conclusion of a category, its RR encoding and its syntactic type are passed to the second stage. Thus the types of items received at the second stage are: noun phrases, verbals, prepositional phrases modifying verbs, complementizers (which introduce complete embedded clauses), relative pronouns (which introduce relative clauses), and conjunctions. The second stage uses the syntactic information to incrementally attach the first-stage pieces into higher level structure, yielding clauses. In the remainder of this article, we concentrate on how these building blocks are combined in the second stage of analysis.⁸

2.2.2 RR Processing Units

We assume that the RR encoding of a sentence is generated clause by clause, to minimize the amount of information stored in WM. In addition to the temporal representation of

⁸This two-stage approach is reminiscent of the dynamic programming schemes proposed in symbolic, context-free processing systems. These systems decouple the processing of the internal details of a phrase from the determination of the hierarchical position of the phrase in the tree. Intuitively, this captures the context-free insight that a category is likely to have the same internal structure irrespective of where it is attached in the tree (Sheil, 1976). Given this, we save processing resources if reprocessing the internal details of the category is not part of restructuring the category as a whole within a tree.

syntactic structure, WM also encodes other information needed for the parsing process, such as the RR encoding of the current clause being processed (denoted **CurRR**), and the RR encoding of the entire sentence (denoted **TotRR**). When the current clause is complete, **CurRR** is integrated into **TotRR**. At the conclusion of the sentence, **TotRR** contains the RR encoding of the thematic tree.

In order to incorporate an item into an RR encoding, its enclosing scope must be determined. We use **CurSc** to denote the enclosing scope within the current clause; each NP is bound to **CurSc**, and the result is merged with **CurRR**. We use **TotSc** to denote the enclosing scope for the current clause as a whole. When a clause is complete, **CurRR** is bound to **TotSc** and this result is merged with **TotRR**. This integration of the current clause is denoted *chunking*.

2.2.3 An Example

As each item is received from the first stage of RR encoding, it is attached according to the basic algorithm in Figure 1.⁹ Table 1 presents the processing of the following sentence:

12. Sue likes the vase which Joe bought.

At the conclusion of the sentence :

$$\mathbf{TotRR} = \mathbf{sue} + \mathbf{likes@(\mathbf{the} + \mathbf{vase})} + \mathbf{likes@C@(\mathbf{joe} + \mathbf{bought@Gap})}$$

C is a predefined item indicating an embedded clause which is not a verbal or adverbial complement, such as a relative clause. Since *bind* distributes over *merge*, this is equivalent to:

$$\mathbf{sue} + \mathbf{likes@(\mathbf{the} + \mathbf{vase} + \mathbf{C@(\mathbf{joe} + \mathbf{bought@Gap}))}$$

Placing the encoding of the RC within the enclosing scope **likes** attaches it to the NP which is also in that enclosing scope, namely **the + vase**. Upon encountering the RC following *the vase*, it is not necessary to alter the existing encoding of that NP in order to convert it into an NP modified by an RC. Rather, merging of new information is all that is required. Thus the specific form of the RR encoding allows monotonic, incremental construction of the thematic tree.

Once categories are bound to each other and RR encoded, they are not directly decomposable, but operations can apply to the vector as a whole. For example, **CurRR** can merge as a unit to **TotRR**.

In addition to the processing outlined here, there is also syntactic and semantic featural integration. For instance, the subject must match the verb's syntactic features, such as number and person. Semantic features must also match; for example, if the verb requires an animate Agent, the subject must specify an animate entity. The details of these integrations are beyond the scope of the article. However, we do assume that such information must be available, and that the RR encoding of an item allows access to this information.

⁹This algorithm assumes that the verb is in the active voice. The passive voice is considered in Whitney and Weinberg (in preparation).

```

/* Initialize*/
set WM variables to empty

/* process input */
for each item x
  if (current clause is complete) /* chunk */

    /* integrate current clause */
    TotRR = TotRR + TotSc @ CurRR
    CurRR = empty

    /* integrate current scope */
    TotSc = TotSc @ CurSc
    CurSc = empty
  end if

  if start of new clause /* branch */

    /* integrate new scope */
    if (x is a relative pronoun) TotSc = TotSc @ C
    else (if x is an adverb)    TotSc = TotSc @ x
  end if

  /* integrate x itself */
  if (x is a verb or thematic role) CurSc = x
  else if (x is an NP or PP) CurRR = CurRR + CurSc @ x

end for

```

Figure 1: Basic algorithm for generating the RR encoding of a sentence having only right-branching clauses.

x	CurSc	CurRR	TotSc
sue		sue	
likes	likes	sue	
the + vase	likes	sue + likes @ (the + vase)	
which			likes @ C
joe		joe	likes @ C
bought	bought	joe	likes @ C
Gap	bought	joe + bought @ Gap	likes @ C

Table 1: Values of selected WM items after each item **x** is processed from sentence (12). The relative pronoun *which* starts a new clause, giving **TotRR** = **sue + likes @ (the + vase)**. The **Gap** indicates that the head NP of the relative clause, *the vase*, functions as the object of *bought*, in addition to its role as the subject of the main clause. Following the processing of **Gap**, chunking is invoked, yielding the final value of **TotRR** given in the text.

2.2.4 Algorithm enhancements

It is not always the case that a clause following a verb is part of its argument. Consider:

13. Mary kissed Bill when Joe won.

Here a main clause is followed by an adverbial clause. The enclosing scope of the adverbial should indicate that the attachment point is outside of the verb phrase.¹⁰ The desired RR encoding is:

Mary + kissed@Bill + when@(Joe + won@E)

In this case, **kissed** should not be transferred from **CurSc** to **TotSc** after *when* is encountered. Thus, when a new clause is initiated with an adverb, **CurSc** is erased without being incorporated into **TotSc**.

Now consider a preposed adverbial clause, such as:

14. When the vase fell, Mary cried.

Here the adverbial clause does not interrupt any dependencies in the main clause, so the RR encoding of the sentence can be generated directly from the input. However, at the conclusion of the adverbial clause, a higher level clause is resumed. Therefore **TotSc** should be erased at this point, since **when** and **fell** are not part of the main clause.

However, if the preposed adverbial were inside a right-branching clause, as in:

15. John said that when the vase fell, Mary cried.

setting **TotSc** to empty following the adverbial would give the wrong result, because the main clause’s verb (**said**) would be erroneously erased, preventing the complement *Mary*

¹⁰Syntactic tests into possible co-referents for pronouns in the adverbial clause show us that the adverb should be attached outside of the verb phrase in what linguists refer to as an adjunction structure.

cried from being attached correctly. Therefore, we divide **TotSc** into two scoping variables: **RgtSc** holds the enclosing scope generated by purely right-branching clauses, and **PreSc** holds the enclosing scope generated by a preposed clause (including right branches from such a clause). This allows the enclosing scope generated by a preposed clause to be deleted without affecting the enclosing scope generated by higher, right-branching clauses. **TotSc** now acts as a switch between these scoping variables: depending on **TotSc**'s setting, binding and erasing operations apply either to **RgtSc** or to **PreSc**. When **CurRR** is chunked, it is bound to both scoping variables. We illustrate this type of processing with an example in section 3.2.1.

The point of this section has been to show how we handle simple cases, generating an output representation appropriate for characterizing linguistic relations (dominance, argument versus adjunct relations, phrasal constituency, etc.) If natural languages consisted exclusively of right-branching (or preposed), unambiguous structure, this representation would suffice, but we now turn to cases which call for enrichment of our mechanisms.

3 Center-embedded structure: Using the WM lists

If a clause is center embedded, the above processing runs into difficulties. In this case, it is not possible to complete the encoding of one clause before starting a new clause. We illustrate this problem with the sentence:

16. John said that when the vase which Sue bought fell, she got very upset.

Here the preposed adverbial is embedded within a sentential complement, and a relative clause occurs before the last argument of the adverbial. If the above processing for a new clause were initiated at *which*, we would have:

TotRR = John + said@when@(the + vase)

In order to interpret the subject-verb relations correctly, it is necessary to remove the material that intervenes between subjects and predicates from processing, and to retrieve the subject. Thus, at *fell*, we need to re-access the subject of the higher clause, *the vase*, so that it can be integrated with that verb. However, here that subject is embedded within **TotRR**, and is inaccessible as a separate component.

This poses a dilemma for a representation that incrementally builds RR encodings because the price for the distributed, compact representation of left context is that it is not easily decomposable. For right-branching or preposed cases, we can refer to non-decomposed pieces in terms of clausal chunks, which can be completed in strict left-to-right order. Since this is not possible for center-embedded dependencies, we need another method of representation that allows access to portions of left context. We accomplish this with a temporal encoding. This allows segmentation of pieces of structure so that they may be referred to individually later. These segments are themselves comprised of RR-encoded structure.

3.1 Temporal representation

The temporal encoding provides a parallel representation of the current clause(s) being processed. If there are incomplete dependencies in the current clause when a new clause is

encountered (i.e., the new clause is center-embedded), RR encoding of the center-embedded clause replaces that of current clause. At the conclusion of the center-embedded clause, the temporal encoding is used to re-access and RR encode the interrupted clause. Thus, the temporal encoding serves as Working Memory, retaining information about incomplete clauses. As discussed in Whitney and Weinberg (in preparation), the temporal encoding also subserves reanalyses of ambiguous sentences.

In order to support RR encoding, WM must encode syntactic structure. We start by specifying the mapping of hierarchical structure onto a temporal representation. Next we discuss how this representation is created during processing. Then we are in position to specify in more detail how the temporal representation is used during the processing of center-embedded structures.

We assume that the basic unit of representation is a list. A list is straightforward to encode temporally: the first item fires first, the next item fires next, and so on. We assume that this firing rides on a carrier wave (an oscillatory cycle). The carrier wave provides a reference frame (i.e., establishes what is “first”), and allows the list sequence to repeat itself over time, providing a memory. The motivations for, and details of, this type of representation are presented in section 4.2.

We propose that syntactic structure is encoded by employing two separate lists - one for noun phrases and other complements, and one for verbs and other verb-related predicates. The lists are synchronized so that items which are in the same position in different lists fire at the same time. The relationship between an NP and a verb is encoded by their relative firing times. Subjects and adjuncts fire asynchronously with their verbs (with subjects firing prior to their verbs in English), while objects fire synchronously with their verbs. For example, the WM representation of

17. Sue loves Bill.

is:

sue		bill	
E		loves	

The top row is the noun list, and the bottom row is the verb list. Each temporal slot is delimited by a vertical bar. The filler item **E** occupies the first slot on the verb list, to establish the proper firing relationships. The first items in each list fire together, then the next items, and so on.

As indicated above, the list encoding is structural; it does not directly encode thematic roles. Thus, a subject fires with the **E** item, and a verb fires with its first object. If a verb does not have an explicit object, the object slot on the verb list must still be occupied; it is filled with the **E** item. If the verb has a second object, the second object fires with another copy of the verb. A PP which modifies a verb is recorded on the noun list, and fires with a thematic-role specifier on the verb list.

Each item on the lists is accompanied by a tag field, which is used to record additional information pertaining to that item. Recall that the first stage of processing returns syntactic information along with the RR encoding of a phrase. The syntactic information specifies the type of the phrase; this type determines which list the phrase is Appended to, and is used to update the internal state of the parser. When the lists are used to generate an RR

encoding, it is necessary to access this syntactic information, so that the parser can return to the correct internal state. This information is stored in the tag field. We assume that any other auxiliary information needed for parsing is also stored in this field. For example, the temporal encoding of the preposed adverbial in (16), before processing *fell*, is:

[Fnp]	[Fnp]	[Gap]
the + vase	sue	Gap
E	E	bought
[E]	[RC]	[Vb]

The values of the respective tag fields are indicated in brackets. **Vb** denotes verb and **Fnp** indicates that the item is a full noun phrase. We assume that syntactic information returned by the first stage includes information about the type of a NP. For instance, NPs that are pronouns need to be identified so that referent processing can be invoked. The **RC** tag specifies that the item is the start of a relative clause. Because the verb list only records predicates related to verb arguments and adjuncts, the **RC** predicate is stored in the tag field. For compactness, we will omit the tag fields if they are not relevant to the processing at hand.

Next we consider how this encoding is created. When an item is added (Appended) to a list, it comes to occupy the earliest empty temporal slot in that list. (The underlying mechanism is discussed in section 4.2.) This allows the temporal representation to be created by sequentially Appending items in order. In general, each argument or predicate is Appended to its respective list as it is received. A subject also induces an **E** to be Appended to the verb list. For example, for (17), **sue** is Appended to the noun list, and **E** to the verb list. Then **loves** is Appended to the verb list, and **bill** to the noun list. Since each item comes to occupy the earliest empty position, and since the two lists are synchronized, **sue** comes to fire synchronously with **E**, and then **bill** fires with **loves**. The temporal representation always carries a parallel encoding of the current clause; when chunking occurs, the list encoding of the clause is deleted.

3.2 Processing Center-Embedded Structures

Now, we will see how the addition of the temporal encoding allows us to handle the problem of center-embedding. When a center-embedded clause is encountered, the current (higher) clause is not chunked: it is not deleted from the lists, nor is it integrated into **TotRR**. **CurRR** and **CurSc** are set to empty, so that they encode will only encode the embedded clause. When the embedded clause is complete, its temporal representation is deleted from the lists and is replaced with its RR encoding.¹¹

The information on the lists is then used to reinstantiate the RR encoding of the higher clause; this process is denoted *WM-RR encoding*. During WM-RR encoding, the lists are accessed slot by slot. As usual, predicates update **CurSc**, and arguments are incorporated by binding to **CurSc** and merging the result with **CurRR**. Following WM-RR encoding, **CurRR** then holds the encoding of the higher clause, and its processing can be resumed.

¹¹Readers familiar with shift/reduce parsing will recognize this as a reduce operation.

3.2.1 Example

We illustrate the use of the lists and of multiple scoping variables for (16), repeated here.

18. John said that when the vase which Sue bought fell, she got very upset.

The following example illustrates several points. We initiate clauses without necessarily indicating that they are dominated by previous material through use of the scoping variables. We handle interrupted dependencies (such as that between *said* and its complement clause *she got very upset* and between *the vase* and *fell*) by (1) refraining from integrating the elements into **TotRR**, and (2) processing and then RR encoding the intervening elements, and (3) retaining the initial elements of the interrupted dependencies as *separate* units in the list representation. In order to handle the enclosing scope of center-embedded clauses, we add the scoping variable, **CntrSc**.

- Initially, **TotSc** is switched to **RgtSc**, and the WM variables and lists are empty.
- After *John*, **CurRR** = **john**.
- After *said*, **CurSc** = **said**.
- *that* indicates a right-embedded clause to be bound to **said**. The normal chunking operation is invoked, giving:

TotRR = **John**; **TotSc** → **RgtSc** = **said**; **CurSc**, **CurRR** = **empty**

and the main clause is deleted from the lists, leaving them empty.

- The adverbial *when* indicates a preposed clause. **TotSc** is switched to **PreSc**, giving:

RgtSc = **said**; **TotSc** → **PreSc** = **when**

- At *which*,

CurRR = **the + vase**; **CurSc** = **empty**

Since *which* starts a center-embedded clause, **CurRR** is set to empty, without integrating it into **TotRR**. This is necessary to allow the head of the relative clause *the vase* to retain its identity so that it can later serve as a subject. **TotSc** is switched to **CntrSc**, giving:

RgtSc = **said**; **PreSc** = **when**; **TotSc** → **CntrSc** = **C**

- At *fell*, the list encoding is:

[Fnp]	[Fnp]	[Gap]
the + vase	sue	Gap
E	E	bought
[E]	[RC]	[Vb]

and **CurRR** encodes *Sue bought*. Since the center-embedded clause is complete, it is chunked. This entails removing the temporal encoding of the clause from the lists, and replacing it with its RR encoding. Thus, when chunking a center-embedded clause, **CurRR** is bound to **CntrSc** and Appended to the lists, rather than being incorporated into **TotRR**. This gives:

$$\begin{array}{l|l} [\mathbf{Fnp}] & [\mathbf{Chunk}] \\ \mathbf{the} + \mathbf{vase} & \mathbf{C} @ (\mathbf{sue} + \mathbf{bought} @ \mathbf{Gap}) \\ \mathbf{E} & \mathbf{E} \\ [\mathbf{E}] & [\mathbf{E}] \end{array}$$

Since we are returning to a higher level clause, the contents of **TotSc** are erased, and then **TotSc** is switched to **PreSc**, giving:

$$\mathbf{RgtSc} = \mathbf{said}; \mathbf{TotSc} \rightarrow \mathbf{PreSc} = \mathbf{when}; \mathbf{CntrSc} = \mathbf{empty}$$

In order to resume the interrupted clause, WM-RR encoding is invoked, producing the RR encoding of the information on the lists. Afterwards,

$$\mathbf{CurRR} = \mathbf{the} + \mathbf{vase} + \mathbf{C} @ (\mathbf{sue} + \mathbf{bought} @ \mathbf{Gap})$$

Thus the lists allowed recovery of the subject of the adverbial clause.

- At *she*, **CurRR** encodes *the vase which Sue bought fell*. The adverbial is complete, so **CurRR** is chunked, giving:

$$\mathbf{TotRR} = \mathbf{John} + \mathbf{said} @ \mathbf{when} @ (\mathbf{the} + \mathbf{vase} + \mathbf{C} @ (\mathbf{sue} + \mathbf{bought} @ \mathbf{Gap}) + \mathbf{fell} @ \mathbf{E})$$

Since we are resuming a higher level clause, the contents of **TotSc** are erased, and then **TotSc** is switched to **RgtSc**, giving:

$$\mathbf{TotSc} \rightarrow \mathbf{RgtSc} = \mathbf{said}; \mathbf{PreSc} = \mathbf{empty}; \mathbf{CntrSc} = \mathbf{empty}$$

The adverbial is deleted from the lists, leaving them empty. Now, **TotRR** encodes the sentence so far, the enclosing scope for the upcoming clause is **said**, and the right-branching complement can be attached in the normal way, as if there had been no intervening material.

In this example, each scoping variable held only a single predicate. This will not necessarily be the case for all sentences, but our mechanisms are adequate for natural language in that the three clausal scoping variables are sufficient to handle any combination of multiple embeddings.

Thus far we have considered preposed, right-branching, and center-embedded clauses. A clause can also be left-branching, such as the sentential subject in:

19. That Mary loves Phil upsets Bob.

At the conclusion of the sentential subject, all items on the lists can be deleted (as for a right-branching or preposed clause), since there are no other clauses on the lists. However, **CurRR** should not be integrated into **TotRR**, since it encodes a subject which must be available for integration with its verb. Therefore, in terms of scoping and integration, a sentential subject is treated like a center-embedded clause.

3.2.2 Deletion from the Lists

When a proposition is complete, it is removed from the lists. Next we consider this deletion in more detail. If it is the only clause in WM, deletion can be accomplished via inhibition of all active items, emptying the lists. We will denote this a *full delete*. However, if there are other clauses on the lists, only a portion of the list items should be removed (as in the previous example where we needed to remove an RC while retaining its head). This is a much more expensive operation, since it involves identifying the correct items to delete. We will denote this a *partial delete*.

For a partial delete, there must be some way of locating the slot at which to begin removal. There are two possibilities for specifying this deletion point: by its ordinal position in the list, or by its identity. Identification by ordinal position would require addition of a counting mechanism. Such a mechanism is suspect, as many have observed that natural language seems to crucially eschew use of counting predicates.¹² Therefore, we assume that a list item is located by its identity.

Since identification of a deletion point requires supplementary parsing information, deletion points are marked in the tag field. When a center-embedded clause is encountered, its subject is recorded as a delete point by merging **Del** into its tag field. The value of this field is recorded in WM in **DTag**. When it is time to delete the clause, the noun list is searched for a tag field matching **Dtag**. When such a match occurs, the items which just fired and all subsequent items on both lists are inhibited.

So, in the above example, the tag field of the RC's subject is actually **Fnp + Del**. This vector is recorded in **Dtag**. When the RC is complete, the noun list is searched for a tag field matching **Fnp + Del**. This match occurs in the second slot. The items in the second and third slots are inhibited, deleting the RC from WM. Thus, the important point is that the temporal encoding functions like a memory address in symbolic systems. The label on the category indicates where the category began, and where the parser must return after the category's completion.

We assume the temporal representation is a basic mechanism of encoding order information in the brain (Lisman & Idiart, 1995). Therefore, to preserve order information, access to a list should start at the beginning of the oscillatory cycle, so that the first item is read out first, the second item is read out next, etc. Thus a list cannot be accessed at arbitrary points. Due to this general constraint, access to a list in the search for a deletion point starts at the first item, and the search proceeds through the successive items. This restriction will be crucial to our account of various psycholinguistic phenomena.

Having discussed how deletion occurs, we are now in a position to specify the full parsing algorithm, given in Figures 2 and 3. This algorithm allows successful encoding of a wide range of multiple embeddings, as we discuss in section 4.3.

3.3 Head-Last Languages

For unambiguous sentences in English, WM-RR encoding is only required when a clause is left- or center-embedded. However, WM-RR encoding plays a more significant role in head-last languages, such as Japanese. In such a language, predicates follow their arguments; the

¹²Berwick and Weinberg (1984) provide a full review of evidence for this position.

```
/* Initialize */
set WM variables and lists to empty
switch TotSc to RgtSc

/*Process input */
for each item x

    if (x starts an embedded clause)
        if (x is an adverb) CurSc = empty
            Chunk, unless incomplete dependencies
            Branch
        else if (x resumes a higher clause)
            Chunk
        end if

    if (x is a verb or thematic role)
        CurSc = x
        Append x to predicate list
    else if (x is an NP or PP)
        CurRR = CurRR + CurSc @ x
        Append x to argument list
    end if

end for
```

Figure 2: Full RR encoding algorithm, using Chunk and Branch operations specified in Figure 3.

Chunk

```
/* Remove current clause from lists */
if (Dtag is empty) empty lists
else    partial delete starting at Dtag; empty Dtag

/* integrate current clause, current scope */
if (TotSc switched to CntrSc) CntrRR = CntrRR + CntrSc @ CurRR
else    TotRR = TotRR + RgtSc @ PreSc @ CurRR

if (resuming higher clause)
    empty TotSc, CurRR, CurSc
    if (TotSc points to CntrSc) Append CntrRR; WM-RR encode
    if (PreSc not empty) Switch TotSc to PreSc
    else    Switch TotSc to RgtSc
else /* starting embedded clause */
    integrate CurSc into TotSc
    empty CurRR, CurSc
end if
```

Branch

```
/* Record start of clause */
if (lists not empty) Dtag gets tag field of subject

/* Set clausal scope */
if (new clause is preposed)
    Switch TotSc to PreSc
else if (left- or center-branching)
    Switch TotSc to CntrSc
    Integrate x into tag field
end if

/* integrate new scope into clausal scope */
Bind x to TotSc
```

Figure 3: Chunking and branching procedures for the full RR encoding algorithm. In order to handle a right-branching clause within a center-embedded clause, we add the variable **CntrRR**, which accumulates the RR encoding of a center-embedded clause.

basic word order is Subject Object Verb. Therefore, it is not possible to generate the RR encoding directly from the input, since the verb which determines the enclosing scope of the object occurs after that object. Rather, it is always necessary to use WM-RR encoding. Note that the same basic list-encoding algorithm works: Append an **E** to the verb list for a subject, Append the NPs to the noun list, and the verb to the verb list. Although the object and verb appear in a different order than English, the final result is the same. The object is the second item to be added to the noun list, and the verb is the second item to be added to the verb list. Therefore, they will wind up firing synchronously. For example, a sentence of the form:

20. NP1 NP2 V

generates the list encoding:

$$\begin{array}{c|c} \mathbf{NP1} & \mathbf{NP2} \\ \mathbf{E} & \mathbf{V} \end{array}$$

Thus the basics of list processing do not vary with head position; they are universal. WM-RR encoding can then be used to produce the representation of the sentence.

4 Computational Demonstrations

Thus far we have discussed how the RR encoding and WM interact in processing a sentence. This section deals with demonstrating that the proposed representations are neurobiologically plausible, and that the proposed algorithm operating on those representations is powerful enough to handle the variety of multiply embedded structures. First we discuss how the RR and WM encodings are mapped onto the neural level, supported by computational demonstrations. Then, abstracting away from this level of representation, we present simulations of sentence processing. We then compare our approach with another type of distributed representation.

4.1 RR Encoding

RR encoding is compatible with a neural framework because it is a distributed representation, and the combinatory operations rely on local processing between nodes (i.e., vector positions). In this demonstration, we have chosen to use the RR encoding proposed by Kanerva (1995) because it uses binary, bitwise operations; it is the simplest of the RR schemes that have been proposed, and is presumably the easiest to implement in neural tissue. We do not necessarily propose that this particular RR encoding scheme is used by the brain. Rather, we use this particular system as an existence proof that an RR encoding is viable for representing sentence structure.

In building up the RR representations, we used the merge and bind operators. First we define these operations, then we discuss how information encoded within a vector is decoded. An item is represented by a random pattern over a large, dense, binary vector, where each bit has equal probability of being a 0 or 1. Merging is implemented as a normalized sum of the constituent vectors, by taking a bitwise majority. That is, at each position, if there

are more 1's than 0's the result is a 1; otherwise it is a 0. Ties are broken deterministically, with an equal probability of giving a 0 or 1 (e.g., by using that position's value in a fixed reference vector). For example:

$$\mathbf{0\ 0\ 1\ 0} + \mathbf{1\ 1\ 1\ 0} + \mathbf{1\ 0\ 0\ 1} = \mathbf{1\ 0\ 1\ 0}$$

Binding is implemented using exclusive-or. That is, at each position, if only one bit is a 1, the result is a 1; otherwise it is a 0. For example,

$$\mathbf{1\ 1\ 0\ 0} @ \mathbf{1\ 0\ 1\ 0} = \mathbf{0\ 1\ 1\ 0}$$

Since composition of two vectors yields a vector in the same representational space, composition entails compression of the constituent vectors, thereby introducing noise. In order to clean up noisy vectors, we assume that there is an item memory which stores the patterns of all base vectors. When presented with a vector, the item memory returns any vector that has a similarity measure above some threshold. Given two vectors, similarity is measured as the fraction of bits that have the same value in both vectors. For unrelated vectors, this measure has the expected value of 0.5. That is, since each bit in a vector has equal probability of being a 0 or 1, the probability that the corresponding bit in another unrelated vector has the same value is 50%.

It must be possible to retrieve the information stored within an RR encoding. Thus, we also need to define the inverse operators, unmerge and unbind. Since the merge operation yields a vector that is similar to its constituent vectors, this allows unmerging to be performed by comparing the vector to the items in memory. For example, assume $\mathbf{A} = \mathbf{B} + \mathbf{C}$, where \mathbf{B} and \mathbf{C} are base vectors stored in item memory. When \mathbf{A} is compared to item memory, \mathbf{B} and \mathbf{C} each have expected similarity values (w.r.t. \mathbf{A}) of 0.75, while the expected value for every other item in memory is 0.5.

The unbinding operator is the same as the binding operator: bit-wise exclusive-or. When unbinding, it is necessary to know what vector to unbind with. When a predicate is bound to an argument during encoding, that predicate is also bound to a special predefined tag which acts as a "hook", allowing that predicate to be retrieved during decoding. We denote this item \mathbf{P} . After unbinding, the resulting vector is cleaned up by comparing it to memory.

Decoding proceeds as follows. First the vector is compared to item memory. This retrieves the unbound constituent, i.e., the Agent. Then the vector is unbound with \mathbf{P} to retrieve any predicates. The vector is unbound with each of those predicates to retrieve their arguments.

We first illustrate decoding at the conceptual level, then we present a numerical example. Consider the vector:

$$\mathbf{Q} = \mathbf{ann} + \mathbf{loves@P} + \mathbf{loves@joe}$$

where \mathbf{ann} , \mathbf{loves} , and \mathbf{joe} are stored in item memory. Comparing \mathbf{Q} to item memory yields the subject \mathbf{ann} , since it is the only constituent vector that is unbound. The vector \mathbf{Q} is unbound with \mathbf{P} , yielding:

$$\mathbf{Q\#P} = \mathbf{ann\#P} + \mathbf{loves@P\#P} + \mathbf{loves@joe\#P}$$

which is similar to \mathbf{loves} . It is also similar to $\mathbf{ann\#P}$ and $\mathbf{loves@joe\#P}$, but these vectors are not stored in memory and act as noise. Thus, comparing $\mathbf{Q\#P}$ to memory yields \mathbf{loves} . The vector \mathbf{Q} can then be unbound with \mathbf{loves} to yield \mathbf{joe} .

If too many relationships are encoded within a single vector, so much noise could be introduced that it would not be possible to retrieve the base vectors. This can be remedied by storing intermediate results in a short-term memory. The result of an unbinding which is not a base vector can then be cleaned up by retrieving the similar item from short-term memory. We assume that if an argument is itself a clause, its encoding is stored in short-term memory. During decoding, the result of an unbinding is compared to both long-term and short-term memory.

Next we present a numerical example of the encoding and decoding of a sentence containing an embedded clause. Using vectors of dimension 10,000, we encoded the sentence:

21. John told Mary that Bill gave Sue money.

where **john**, **mary**, **bill**, **Sue**, **told**, **gave** and **money** were stored in item memory, along with 3000 distractor items. The encoding of the sentential complement *Bill gave Sue money*:

$$\mathbf{V1} = \mathbf{bill} + \mathbf{Goal}@\mathbf{(sue + P)} + \mathbf{gave}@\mathbf{(money + P)}$$

was entered into short-term memory. The encoding of the entire sentence was:

$$\mathbf{V} = \mathbf{john} + \mathbf{Goal}@\mathbf{(mary + P)} + \mathbf{told}@\mathbf{(V1 + P)}$$

We then decoded \mathbf{V} as follows. (We present all items exceeding a similarity cutoff of 0.52, with the similarity value in parentheses.) \mathbf{V} was compared to item memory to get the Agent. This yielded **john** (0.75). \mathbf{V} was unbound with \mathbf{P} , and the result was compared to memory to get any predicates. This yielded **told** (0.63) and **Goal** (0.62). \mathbf{V} was unbound with **told** to get the Theme, yielding $\mathbf{V1}$ (0.62) and **bill** (0.57). This similarity to **bill** is appropriate because \mathbf{V} contains **told @ bill**; this indicates that **bill** is the Agent of $\mathbf{V1}$. \mathbf{V} was unbound with **Goal**, yielding **mary** (0.62). This process was repeated with $\mathbf{V1}$. The Agent was **bill** (0.75). Unbinding $\mathbf{V1}$ with \mathbf{P} yielded **gave** (0.63) and **Goal** (0.62). Unbinding with **gave** yielded **money** (0.63) and unbinding with **Goal** yielded **sue** (0.63). Thus, it was possible to retrieve the structure specified in the RR encoding of the sentence.

Since the merge, bind, and unbind functions operate on corresponding bits across two vectors, they could be implemented within a neural network using one-to-one connections between the areas over which the input vectors are represented. The unmerge operation requires an auto-associative memory. Recurrent networks employing distributed representations have long been touted as a natural framework for this kind of memory. As such, all of these operations are neurobiologically plausible.

4.2 Working Memory

Next we discuss the underpinnings of the WM lists. Recent EEG experiments indicate that verbal WM employs oscillations in the theta range (about 5Hz), supporting our assumption of temporality. (Klimesh, 1999; Raghavachari, Kahana, Rizzuto, Caplan, Kirschen, Bourgeois, Madsen & Lisman, 2001). It has been suggested that oscillatory activity in the 40 Hz range is related to cognitive processing (Tiitinen, Sinkkonen, Rainikainen, Alho, Lavi-Kainen, & Naatenen, 1993) and that short-term memories are encoded on 40 Hz subcycles of a theta oscillation (Lisman & Idiart, 1995).

We have previously proposed that such 40 Hz subcycles encode the order of letters within a word, and have shown that this model accounts for a wide range experimental data (Whitney, 2001; Whitney & Berndt, 1999). We assume that a similar encoding underlies the realization of a WM list, wherein each temporal slot corresponds to a successive 40 Hz subcycle within a 5 Hz oscillation. Consistent with this assumption, there is a slow and highly frequency-specific increase in theta power as a sentence is being processed (Bastiaansen, van Berkum & Hagoort, 2002).

Initially we assume that each list item is represented by a single cell, and then we discuss vector items. Following the proposal of Hopfield (1995), we assume that all cells undergo synchronous, subthreshold oscillations of excitability; therefore, the magnitude of an input to such a cell determines when firing threshold is exceeded. For a small input, threshold is not exceeded until late in the cycle when the cell's oscillation brings its potential near threshold. For a larger input, threshold is exceeded earlier in the cycle. Thus, the size of an input determines the spike timing relative to the oscillatory cycle. We assume that when a cell fires, it sends inhibition to the other cells. This inhibition disallows other cells from firing for approximately 10-20 ms, creating subcycles within the oscillatory cycle.

It has been proposed that an afterdepolarization (ADP) can maintain short-term memory across oscillatory cycles (Lisman & Idiart, 1995). The ADP is a slowly increasing excitability that peaks at approximately 200ms after spiking, and has been observed in cortical pyramidal cells. As such, the ADP could cause refiring on successive oscillatory cycles in the absence of external input. We assume that once a cell on a list has been activated by external input, its activation is maintained across cycles through the ADP. The slowly increasing ramp of the ADP can preserve the firing order of elements across oscillatory cycles, as demonstrated by a mathematical model (Lisman and Idiart, 1995). For example, the cell firing first (denoted cell A) and the cell firing second (cell B) within one cycle will have the highest and second-highest levels, respectively, of ADP in the next cycle. In the next cycle, these ADP levels interact with the temporal variation in excitability, allowing cell A to reach threshold before cell B. Thus the order of firing of cells A and B is carried from one cycle to the next.

We assume that an item is Appended onto the list by activating it near the peak of the oscillatory cycle; the ADP allows the cell to fire progressively earlier within successive cycles, until it is limited by the trough of the cycle (for the first item), or by inhibition from other cells firing prior to it. We have performed simulations of the neural dynamics of the Append operation based on the Lisman and Idiart (1995) model, described in Whitney (in preparation). 8 items were appended, one per cycle. After 9 cycles, all items fired in the correct sequence within a cycle: items 1-8 fired at times 31, 48, 61, 71, 80, 88, 95, 103, respectively (relative to the start of the cycle).

However, our proposed list items are not single units, but rather large, binary vectors. We posit that a large bank of cells exists for each vector position. A vector is represented by synchronous firing across these banks. All vectors in a list are represented by these same banks of cells. When a position is set to 1, a subset of the cells in that positional bank is activated. Thus, for a 1 in the same position in two different vectors, different subsets of cells are active on different temporal subcycles. If no cells fire in a given bank on a given subcycle, the value of that position is taken to be 0. We assume that each subset is activated stochastically from the cell bank; thus it is possible for a cell that is already representing an item x to be recruited to represent a different item y . However, it is unlikely that all cells

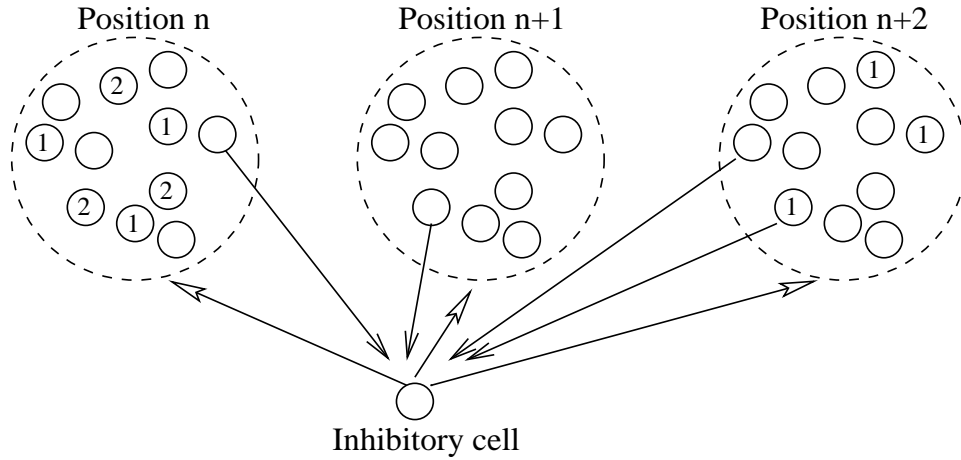


Figure 4: Architecture of the proposed neural realization of a list, shown for a subset of positions. Each position is comprised of a bank of cells. (A much larger number of cells per position is assumed than is shown here.) Each cell sends excitatory output (shown here only for some cells) to the inhibitory cell. The inhibitory cell sends its output to all positional cells. All positional cells are assumed to undergo synchronous subthreshold oscillations. Excitatory input received from an external source activates subsets of the positional cells. Numbers indicate the temporal slot in which a cell fires in the following example. Assume that this list currently contains two items: the first item has values 1, 0, and 1 in positions n , $n+1$, and $n+2$, respectively, and the second item has values 1, 0, and 0 in those positions. During the first subcycle, subsets of cells in positions n and $n+2$ fire. During the second subcycle, a different subset of cells in position n fires.

representing \mathbf{x} will be reassigned. Thus, the activation of a new item can reduce the level of activation of previous items somewhat. The more similar a new item is to an old item, the more active positions are shared, and the more likely it is that active cells from the old item will be redirected to the new item.

When a cell fires, it sends inhibition to all other cells within the same positional bank and across banks, creating a subcycle within the oscillatory cycle. This could be accomplished by having each cell send excitatory connections to a cell that sends inhibitory connections to the banks of cells. Thus, a single list is comprised of the inhibitory cell and its target cells. See Figure 4. Separate lists exist when different inhibitory cells have different sets of target cells.

As discussed in the Introduction, the problem of how to represent a hierarchical structure temporally is a difficult one. There must be a mapping from a two-dimensional structure (a tree) onto one-dimensional structure (time). Given this mapping, there must be a way to initiate the correct firing times. Our proposal for the structure of WM solves these problems. Because the oscillatory cycle provides a reference frame, two items that are activated at different times on different lists can come to fire synchronously. Because the order of firing is maintained across cycles, information can be encoded by the sequence of firing, not just by the synchrony of firing. This allows right-branching structure to be mapped on to successive temporal slots. However, the proposed temporal representation is not a stand-alone encoding. Rather, it is closely integrated with the process of RR encoding.

This allows left-branching and center-embedded structures to be represented within a single temporal slot by RR encoding that structure. These factors allow arbitrary hierarchical structure to be represented in WM.

4.3 Sentence Processing

Having described the details of the representations supporting sentence processing, we now turn to the parsing algorithm itself. We performed simulations of the proposed algorithm on a variety of multiply embedded clauses to demonstrate that such an algorithm is powerful enough to handle the structures found in natural language. Our aim is to show the proposed algorithm is viable at the computational level without committing to any particular neural implementation of the parsing rules themselves; therefore, we implemented these simulations at the symbolic level.

Each input sentence is represented by a sequence of two-character symbols. The first character is alphabetic, specifying syntactic type, and the second character is numeric, distinguishing different instances of the same syntactic type. For example, the input representation of a sentence with a sentential complement containing a preposed adverbial clause having a right-branching relative clause, like:

22. John said that after Mary dropped the vase that Jim bought, she got a new one.

is:

N1 V1 C1 A1 N2 V2 N3 R1 N6 V3 N4 V4 N5

where 'N' specifies an NP, 'V' a verb, 'C' a complementizer, 'A' an adverb, and 'R' an relative pronoun. The output for each sentence is a string specifying the RR encoding of the sentence. For the above sentence, the desired output is:

N1 + V1 @ A1 (N2 + V2 @ N3) + V1 @ A1 @ C @ (N6 + V3 @ Gap) + V1 @ (N4 + V4 @ N5)

We implemented the algorithm given in Figures 2 and 3. A positional variable taking one of four values (0 = before subject, 1 = before verb, 2 = before object, or 3 = after object) was used to determine the branching direction of an embedded clause. The model was tested on a wide variety of sentences containing multiple embeddings of relative, sentential, and adverbial clauses. These inputs are listed in the Appendix. The correct output was generated for each sentence, except those containing doubly center-embedded relative clauses, and one containing a noun complement center-embedded within a center-embedded relative clause. As we discuss in section 5, people also have difficulty with such constructions.

4.4 Connectionist Distributed Representations

Next we attempt to evaluate our model in a broader computational context. In order to demonstrate the value of our RR encoding scheme, it is useful to compare it to other work that uses distributed representations to deal with natural language. We share with these architectures a desire to construct models that are neurobiologically plausible. Most of the other work in this framework is labeled a *connectionist* architecture.

Here we use the term *connectionist* to refer to a particular approach to distributed representations, namely training recurrent networks via a backpropagation algorithm. We are interested in connectionist systems that can encode the hierarchical relationships implied by a string of words; it must be possible to decode this encoding to retrieve those relationships, for they are crucial to the association of sound and meaning that we believe underlies language understanding.¹³

An attempt to present such a system has recently been made (Rohde, 2002). In this work, the meaning of a sentence was represented as a list of propositions. Each proposition had three parts - the action, the role, and the filler. For example, the encoding of:

23. Jim knows Bill loves Sue.

is:

```
(knows, agent, jim)
(knows, theme, loves)
(loves, agent, bill)
(loves, theme, sue)
```

First the system was trained to form a distributed encoding of a list of propositions. Such an encoding was decoded by querying, where two of the three parts of a proposition are specified and the system must fill in the third part. Training was performed by presenting propositions one at a time, and learning to correctly respond to queries about every proposition presented so far. Thus, this part of the system learned to create an RR encoding for a sequence of propositions.

Next the system was trained to produce such an RR encoding for a sentence, as follows. The propositions representing the meaning of a sentence were fed through the system to get their RR encoding. Then the words in that sentence were presented, and a different part of the system learned to produce that RR encoding in response to those words. Following training on a wide range of sentences, the representational ability of the system was tested by presenting a novel sentence, and then querying the resulting RR encoding on each proposition comprising the meaning of that sentence.

While the model showed good generalization abilities for simple sentences, performance rapidly deteriorated for more complex structures. For example, the average error rate for a sentence with six propositions was approximately 10%, according to the less stringent multiple-choice criterion. That is, the response to a query was counted as correct if it was closer to the correct response than to other distractor items. This error rate is per proposition, so the probability of correctly encoding the entire sentence is $(1.0 - 0.10)^6$, about 50%. For eight-proposition sentences, the error rate was about 20%, so the probability of correctly representing the sentence was only about 15%. Error rates on center-embedded structures were particularly high. For a four-proposition sentence having a subject-modifying

¹³Despite the logical necessity of demonstrating that a system can perform this basic representational function, some researchers (e.g., Christensen & Chater, 1999; Elman, 1993; Seidenberg, 1997) have made claims about the ability of connectionist systems to handle natural language without producing such a system. Rather they have produced systems which only learned to predict the next word in a sequence; it was not possible to retrieve any relationships specified by the sequence of words. This clearly does not constitute a demonstration that such a system is up to the real tasks of natural language processing.

RC, error rates were at 25%, so the probability of correctly encoding the sentence was about 30%. Thus, these simulations have not demonstrated that such a connectionist system is capable of developing representations than can handle complex syntactic structure.

In contrast, we have proposed a systematic way of representing syntactic structure in a distributed fashion. It is possible to encode any complex set of relationships, since we have defined combinatory operators that can be recursively applied. Assuming that encodings of sub-structures are saved, this allows arbitrarily large, complex structures to be encoded and decoded without degradation.

5 Complexity

Next we show that the comprehension difficulty of a set of unambiguous sentences can be correlated with natural features of the model. The first case deals with how easily interrupted dependencies can be incorporated into the WM. The second set can be ranked with respect to whether list deletion is initiated at the correct place.

5.1 Cross-Serial Dependencies

Recall that the temporal encoding consists of two lists (a predicate list, and an argument list) that must be synchronized. This encoding across dual lists and the expression of relations in terms of synchrony and temporal order allows a natural account of the difference in perceived complexity between a class of cases in Dutch and German that would otherwise seem to be of comparable difficulty. For nested verbs in Dutch which subcategorize for infinitival complements, all subjects and objects precede the verbs. The first NP is the subject of the first verb, the second NP is the subject of the second verb, etc. For example, a sentence like:

24. Joanna helped the men teach Hans to feed the horses.

is expressed as follows:

25. Jeanine heeft de mannen Hans de paarden helpen leren voeren.
 Joanna has the men Hans the horses helped to-teach to-feed.

The English version of this sentence is right-branching; the RR encoding can be generated directly, without using WM. This is because the verbs precede their objects; thus the verbs can be accumulated in **RgtSc**, to become predicates for the noun phrases which follow, creating clauses that then can be chunked in order. By contrast, the subjects in the Dutch cases must be held as individual units in order to be associated with the correct verbs. However, because we can hold each NP in a separate temporal slot of the WM, we can use the same rules as we used for English. The RR encoding can then be generated from the WM encoding. After processing the NPs, the WM encoding is:¹⁴:

joanna | **the+men** | **Hans** | **the+horses** |
E

¹⁴We assume that the auxillary verb between the first and second subjects is saved elsewhere to be joined with the first main verb.

Now each verb is Appended to the verb list in turn. The result is that each verb comes to fire after its subject:

joanna	the+men	Hans	the+horses
E	has-helped	to-teach	to-feed

We can then read the items off the lists in order, and RR encode them in the manner of right-branching clauses, giving:

joanna + has-helped@(the + man) + has-helped@to-teach@Hans+
has-helped @to-teach @to-feed@(the + horses)

which is equivalent to:

joanna + has-helped@(the + man + to-teach@(Hans + to-feed@(the + horses)))

In contrast, the corresponding dependencies in German result in a set of center- embedded clauses intervening between subjects and predicates. The relevant German sentence is:

26. Johanna hat den Mannern Hans die Pferde futtern lehen helfen.
 Joanna has the men Hans the horses to-feed to-teach helped.

As discussed above, we require that the embedded clause be located and removed from the lists in order to correctly associate the separated subjects and predicates. Verbs cannot be automatically slotted into the correct synchronous locations, as is possible for Dutch.

Because the center-embedded structures require partial deletes, while the cross-serial dependencies don't, our prediction is that cross-serial dependencies should be easier to process than center-embeddings. A study of the relative ease of comprehension for the Dutch versus German versions showed that the Dutch version is easier (Bach, Brown & Marslen-Wilson, 1986), as predicted by TPARRSE.

5.2 Center Embedding

We now turn to the mapping between the WM and RR, which is the second potential complexity-producing operation in our architecture. Recording deletion points by tagging, as required to remove material interrupting the dependencies in a higher level clause, raises the possibility of ambiguity; two different slots could have the same tag. We propose that such ambiguity underlies the difficulty of processing some doubly center-embedded clauses in English.

5.2.1 Relative Clauses

A sentence with one RC center-embedded inside another (RC/RC) is very difficult to understand, for example:

27. The vase that the man who Sue dated bought fell.

The lists up to *bought* are:

[Fnp]	[Fnp + Del]	[Fnp + Del]	[Gap]
the+vase	the+man	sue	Gap
E	E	E	dated
[E]	[RC]	[RC]	[Vb]

with **Dtag** = **Fnp + Del**. We refer to the subjects of the outer RC and the inner RC as *N2* and *N3*, respectively. Since the search for **Dtag** starts at the beginning of the list, it will match at *N2*. However, this is the incorrect point to begin deleting; deletion of the inner RC should start at *N3*. Thus, deletion of the inner RC causes *N2* to be erroneously removed from processing, giving:

the+vase	C @ (sue + dated @ Gap)
E	E

When two more verbs are received, the sentence doesn't make sense because there is only one subject remaining on the lists.

This deletion error explains the surprising result that an ungrammatical sentence can be judged as acceptable as grammatical one. An ungrammatical version of an RC/RC, where the second verb has been dropped, is as acceptable as, or more acceptable than, the grammatical version (Frazier, 1985; Gibson & Thomas, 1999), for example:

28. The vase that the man who Sue dated fell.

Dropping the first or third verb does not yield equally acceptable results. For example,

29. The vase that the man who Sue bought fell.
The vase that the man who Sue dated bought.

We dub this the *V2-drop* effect. This effect is predicted by the proposed processing. Since *N2* has been erroneously removed from the lists, removing the corresponding verb from the sentence makes it seem better: following deletion, only the main verb is received, and is associated with the one subject remaining on the lists, the main subject.

If the difficulty of an RC/RC arises from ambiguity of delete tags, making the tags for *N2* and *N3* different from each other should alleviate the problem. Indeed, if *N2* is an indexical (first- or second-person) pronoun, a double center-embedding seems easier than usual, as demonstrated in off-line complexity ratings (Gibson, 1998; Warren & Gibson, 2002a). For example, consider:

30. The vase that the man who I dated bought fell.

We dub this the *N3-type* effect.

Recall that syntactic type is recorded in the tag field. Assuming this, when *N3* is a pronoun, its tag field includes a pronoun tag, rather than a full NP tag. Thus, *N3*'s tag field differs from *N2*'s. Since **Dtag** records the tag field of the most recent delete point (i.e., *N3*'s), and that field is unique, deletion of the inner RC proceeds correctly. Thus the sentence is processed correctly and seems easier.

This indicates that the V2-drop and N3-type effects should interact, since N2 is not erroneously deleted when N3 is a pronoun. We examined this hypothesis in a self-paced reading study (Whitney, in preparation). In the region following the final verb, when N3 was a name, there was an advantage for the ungrammatical condition (i.e., reading times were longer in the grammatical condition than in the ungrammatical condition); when N3 was a pronoun, there was no advantage for the ungrammatical condition. Thus V2-drop was felicitous when N3 was a full noun phrase, but not when it was a pronoun.¹⁵

Next we consider a center-embedded RC within a right-branching RC, for example:

31. I like the vase that the man who Sue dated bought.

This sentence contains the same sequence of words (*the vase that the man who Sue ...*) that causes the difficulty in the double center-embedded example (27), yet processing proceeds correctly. Since the outer RC is right-branching with respect to the main clause, chunking occurs when that RC is encountered. During chunking, the lists are cleared. Therefore, the subject of the outer RC falls into the first temporal slot, so that RC can be removed via a full delete of the lists.¹⁶ Thus, only the subject of the inner RC is tagged with **Del**, so **Dtag** is unique. For example, the lists up to *bought* are:

[Fnp]	[Fnp + Del]	[Gap]
the + man	sue	Gap
E	E	dated
[RC]	[RC]	[Vb]

with **TotRR** = **I + like @ (the + vase)** and **RgtSc** = **like @ C**. This predicts that dropping the outer RC's verb should not be felicitous, unlike the doubly center-embedded case. For example, consider:

¹⁵In our self-paced reading study, we tested first-person pronouns, and third-person pronouns having a referent. These two types of pronouns grouped together in terms of the felicity of V2-drop. However, in off-line complexity judgements, a third-person pronoun without a referent was found to be as difficult as a full NP, while a third-person pronoun with a referent was found to be intermediate in difficulty between an indexical pronoun and a full NP (Warren & Gibson, 2002a). So, in terms of off-line complexity ratings, first- and third-person pronouns do not group together. Thus, overall complexity and V2-drop felicity are not in a one-to-one correspondence. We predict that V2-drop for a third-person pronoun without a referent should group with the other pronouns. We suggest that the increased off-line complexity rating for third-person pronouns (with respect to indexicals) reflects increased processing incurred by searching for a referent (and perhaps not finding one), and does not reflect difficulty caused by incorrect deletion from the lists. (An indexical does not require a search for a referent because its referent is predetermined: the speaker for the first-person pronoun, and the listener for the second-person pronoun.)

¹⁶One might ask though, how the parser retains the identity of the relative clause as part of a noun phrase, once chunking has occurred. We assume mechanisms that are needed independently in the grammar. As has been pointed out many times, while clauses normally function as arguments to verbs, relative clauses and noun-complement clauses (see below) are semantically predicates. A restrictive relative functions like an adjective in asserting that a restrictive property applies to the head of the noun phrase. The standard semantic mechanism for representing this is lambda abstraction, which applies when a predication relation is represented (usually by coindexation between the head and the relative clause) in the syntax. (Chierchia & McConnell-Ginet, 1990). We assume that this coindexation allows the relative to encode the person, number and gender features of the noun it is coindexed with. It would not assume the deletion feature, since the category would already be deleted.

32. I like the vase that the man who Sue dated.

Indeed, Gibson and Thomas (1999) found that such a sentence is less acceptable than the grammatical version.

5.2.2 Noun Complements

A noun complement (NC) is complete clause (i.e., does not contain a gap), and can only follow an NP which has the semantics of a proposition, for example:

33. The claim that Mary made dynamite upset Bob.

An NC does not modify its head NP, as does an RC; rather, it is equivalent to its head NP. This is shown by the ability to use the head and the complement on either side of an identity statement.

34. The claim is that Mary made dynamite.

Therefore an NC following the subject NP is itself the subject, so such an NC is equivalent to a left-branching, sentential subject. We assume that the default method of chunking at a clause boundary is pursued wherever possible. Given the semantic equivalence of a noun complement and its head (which is not the case for the relative clause), we assume that this default strategy can be adopted for noun complements. Therefore, chunking occurs after the head, and the NC is processed like a sentential subject. Thus at *that* in (33), **the + claim** is transferred to **TotRR**, and **TotSc** is switched to **CntrSc**. At *upset*, the NC is chunked. Now the NC functions as the subject of the sentence. Since a chunked clause has the same syntactic and semantic features as an NP which can take a NC (namely those of a proposition), the chunked NC is successfully integrated with the verb. In contrast, a subject NP modified by an RC cannot be chunked prior to processing the RC, because the subject NP would be removed from processing, but the features of a chunked clause would be inappropriate. Note, however, that if a potential NC turns out to be a RC, as in:

35. The claim that Mary made upset Bob.

this is not a problem, because the features of a proposition are still appropriate for replacing the subject NP *the claim*.

This NC processing explains some phenomena associated with the complexity of RC and NC nestings. An NC which contains an RC (NC/RC) seems easier than a RC/RC (Gibson, 1998; Gibson & Thomas, 1997), for example:

36. The fact that the manager who the supervisor hired lost the records bothered the intern.

Since the NC's head NP is chunked and the NC starts at the first slot, there is no **Del** tag for the NC. So the lists up to *lost* are:

[Fnp]	[Fnp + Del]	[Gap]
the + manager	the + supervisor	Gap
E	E	hired
[NC]	[RC]	[Vb]

Since **Dtag** (= **Fnp** + **Del**) is unique, the RC is correctly deleted, and the sentence is correctly processed.

Intuitively, an NC/NC combination seems relatively easy to understand, for example:

37. The claim that the fact that Clinton lied is irrelevant angered Sue.

Here too, the outer NC starts at the first slot, so it is not tagged for deletion. Therefore, **Dtag** is unique and the sentence is correctly processed.

In contrast, a RC/NC seems very difficult (Gibson, 1998; Gibson & Thomas, 1997), such as:

38. The supervisor who the fact that the intern lost the records bothered quit.

In this case, the NC's head NP cannot be chunked, because it is within a center-embedded relative clause. Therefore, the start of the NC must be marked with a **Del** tag. So the lists up to *bothered* are:

[Fnp]	[Fnp + Del]	[Fnp + Del]	[Fnp]
the + supervisor	the + fact	the + intern	the + records
E	E	E	lost
[E]	[RC]	[NC]	[Vb]

Here the tag fields for N2 and N3 are identical, as for an RC/RC. Therefore deletion of the NC starts at the second slot, and the sentence is incorrectly processed.

Recall that a potential NC which turns out to be an RC does not cause processing problems, because the features of a chunked clause are still appropriate. So, a potential NC/RC which turns out to be an RC/RC is processed like the easier NC/RC. Therefore, such an RC/RC should seem easier to understand than an unambiguous RC/RC. For example, consider:

39. The claim that the guy who Sue is dating made was false.

Intuitively, this sentence seems relatively easy, in line with our analysis.

In summary, we propose that the perceived ease or difficulty of double embeddings depends on the process of deletion. If deletion removes necessary information, the sentence is perceived as unacceptably difficult. Of course, we do not think that this is sole determiner of perceived complexity. Other factors are also at work, such as degradation of WM. Because we propose an explicit model of WM, we can also account for effects relating to its degradation. Recall that all items in a list draw cells from a common pool. We assume that activating a new item entails randomly activating a subset of cells from that pool. A cell that is already active in representing one item could be “stolen” to represent a new item. In this way, a new item can reduce the total activation of a previous item. The more similar a new item is to a previous item, the more degradation occurs. The temporal nature of the encoding may also have effects. Items are separated by firing times. If timing drift occurs, some cells representing one item could end up firing with an adjacent item. This will make adjacent items more similar to each other, and more difficult to differentiate. Thus, the degradation of an item in WM increases with the more items that are added after it, the more similar those items are to it, and the more proximal similar items are to it. We assume that such

degradation is also related to perceived complexity. This is consistent with investigations on stacked NPs in Japanese. For a fixed number of NPs, perceived complexity increased as the number of similarly case-marked NPs increased; for a fixed number of similarly case-marked NPs, complexity increased as their proximity to each other increased (Lewis & Nakayama, 2001).

The degradation of WM is also consistent with the proposal that complexity increases as the distance between two items that must be integrated increases (Gibson, 1998, 2000). However, as we discuss below, this proposal by itself is not sufficient to account for the complexity phenomena associated with double embeddings.

5.3 Related Work

Our analysis of complexity is based on the similarity of tags at deletion points. Previous proposals have also focused on similarity as being a source of problems. Closely related to our proposal, it has been suggested that a stack-based parser might confuse two stacked instances of the same category (Miller & Chomsky, 1963). Lewis (1996) has proposed that the parser cannot maintain more than two instances of unattached items indexed under the same syntactic category, due to interference in working memory. However, approaches at this level cannot account for the relative ease of NC/RC and NC/NC embeddings. TPARSSE can account for these cases because our notion of similarity is at a lower level.

In developing the Dependency Locality Theory (DLT), Gibson and colleagues have been responsible for identifying and studying a range of complexity phenomena, including the difference between an NC/RC and an RC/NC, and the V2-drop and N3-type effects (Gibson, 1998; Gibson & Thomas, 1997, 1999; Warren & Gibson 2002a). We feel that any theory of complexity should account for these important phenomena.

The DLT theory is based on the idea that complexity increases as the distance increases between two things that must be integrated together in the syntactic tree (Gibson, 2000). Distance is measured as the number of new discourse referents that intervene, where new discourse referents are tensed verbs, and NPs that are not indexical pronouns.¹⁷ Integration cost is taken to increase with distance because the activation of the first entity is taken to decrease as activation is redirected to new discourse referents; thus more energy is required to reactivate the first entity during integration. A cost of 1 Energy Unit (EU) is generated for each intervening discourse element, and for generating the new discourse referent itself. Perceived complexity corresponds to maximal integration cost. For example, for an RC/RC construction such as :

40. The vase that the man who Jen dated bought fell.

the highest cost occurs at the verb *bought*, which has a cost of 7 EUs: 1 EU for its construction + 2 EUs for attachment to *man* (across *Jen* and *dated*) + 4EUs for coindexing the gap following *bought* with the relativizer *that* (across *man*, *Jen*, *dated*, and *bought*). It is proposed that this high cost corresponds to the unacceptability of such a structure.

An RC/NC has a larger maximal cost than an RC/RC (due to integrating across an explicit object in the NC). However an NC/RC has a lower cost than an RC/RC because a

¹⁷The discourse is presumed to always include a speaker and a listener, so pronouns referring to either do not introduce a new referent.

long-distance integration of a gap across an embedded clause is not required. This accounts for the difference in difficulty between an RC/NC and NC/RC (Gibson, 1998).

Under the DLT's assumption that an indexical pronoun does not introduce a new discourse referent, integrating across such an entity does not generate any cost. This accounts for the N3-type effect. However Warren and Gibson (2002b) tested the discourse-referent hypothesis and failed to show the predicted results. In that study, subjects read sentences in which an object-extracted RC modified the main subject. The subject of the RC was a definite NP. Whether or not this NP had a referent was manipulated in the context prior to the sentence. The presence or absence of a previous referent affected reading times at the RC's verb, but not at the main verb. Thus an NP which adds a new discourse referent incurs a local cost (at its own verb), but does not affect the cost of processing further downstream (at the main verb), contrary to the DLT's prediction. A local effect of discourse-referent processing could still potentially account for the N3-type effect in off-line complexity ratings for the sentence as a whole. However, it can't account for the interaction of N3-type with the felicity of V2-drop, since this is a non-local effect.

Moreover, the DLT can not account for the V2-drop phenomenon at all. An earlier version of the DLT, the SPLT (Gibson, 1998), proposed that complexity corresponds to the storage cost of syntactic *predictions*, not integrations. Under that metric, it was proposed that the parser drops the prediction for the outer RC's verb, due to high memory costs. However, an assumption underlying the SPLT was contradicted by experimental evidence (Gibson, 2000; Gibson, Desmer, Watson, Grodner & Ko, submitted); the SPLT was transformed into the DLT, where prediction cost is constant, and integration cost increases with distance. Hence, under the DLT, prediction cost cannot explain V2-drop, since the prediction cost for the outer RC and the inner RC are the same. While it's true that the outer RC's verb has the highest integration cost, this cost is incurred *after* the verb is encountered, and thus cannot account for dropping the prediction of that verb before it occurs. Accordingly, integration cost at V2 is independent of whether the outer RC is center-embedded, as in (27), or right-branching, as in (31), but V2-drop is felicitous only when the outer RC is center-embedded (Gibson & Thomas, 1999). Thus, integration cost cannot account for the V2-drop effect either.

Furthermore, the DLT makes the wrong prediction about complexity in some important cases. The RC/RC's high cost results from the summation of the integration costs for the second verb and the first RC's gap. However, if these costs are decoupled, as in the following sentence:

41. The woman who the man who Sue dates flirted with hit him.

the complexity is still very high, while the maximal integration cost is lower. Here the intransitive verb *flirted* signals that the gap for the first *who* is not in the object position. So the integration cost of *flirted* is only 3EUs. The integration cost of the gap following *with* is 4EUs, and the integration cost of *hit* is 5 EUs. Thus the maximal cost is only 5EUs. However a much easier sentence like:

42. The fact that the man who Sue is dating rides a motorcycle scares her.

has a higher integration cost, of 6EUs (at *scares*). In contrast, TPARRSE treats the intransitive RC/RC the same as the transitive RC/RC. In both cases, the tag field on N3 is the

same as N2, yielding incorrect deletion of N2.

The above analysis depends on the assumption that integration of a gap is not attempted following an intransitive verb, as is consistent with studies on filler-gap processing for intransitive verbs (Boland & Tanenhaus, 1991; Sussman & Sedivy, 2001). However, even if it were argued that such an integration is attempted and incurs a cost, this would not help the DLT. This claim would then destroy the DLT account of the difference between an NC/RC and an RC/NC. That account hinges on the assumption that there is no long-distance integration of a gap across the RC for an NC/RC. However, there is evidence that the RC possibility for an NC is evaluated: in a potential NC, a manipulation of the potential filler’s appropriateness as the verb’s object showed an effect at the verb, indicating that the possibility of a gap is actively considered (Pearlmutter & Mendelsohn, 1998). Thus, if it were argued that an integration cost for a possible gap is incurred at an intransitive verb, such a cost would surely also apply to a potential NC’s verb. However, in that case, there would be no difference in integration cost for an NC/RC versus an RC/NC. Nor could it be argued that the possibility of a gap is dropped in a potential NC/RC due to increased complexity; this would incorrectly predict that a potential NC/RC which turns out to be an RC/RC, such as (39), is uninterpretable.

Another incorrect prediction occurs for Japanese. A sentential complement within a sentential complement (SC/SC) of the form:

43. NP-nom [NP-nom [NP-nom V1 Comp] V2 Comp] V3

has its highest integration cost at V3 = 5EUs. An SC of the form:

44. NP-nom NP-dat [NP-nom NP-dat NP-acc V1 Comp] V2

has its highest integration cost at V2 = 6EUs. However, the SC is easier than the SC/SC (Babyonyshev & Gibson, 1999). In contrast, under TPARRSE, the double center-embedding resulting from an SC/SC in Japanese would suffer the same problems as an RC/RC in English, making it more difficult than a single center-embedding.

The underlying assumption of the DLT is inherent to the TPARRSE model - that syntactic representations draw from and share a fixed set of resources; therefore, a new syntactic object can affect the activation of previous ones. However, we have demonstrated that this assumption by itself is not sufficient to explain the phenomena associated with double embeddings. The DLT cannot fully account for the difficulty of double center-embeddings, nor the V2-drop effect, nor the interaction of V2-drop with N3-type. Rather, we suggest that it is also necessary to take into account other aspects of the underlying representation, namely how syntactic objects are removed from WM.

6 Conclusion

Many researchers want to understand how language “works”. Psycholinguists and linguists study language from the top, in terms of a symbolic, cognitive system. Neuroscientists and computational modelers study language from the bottom, in terms of brain regions and networks of neurons. Heretofore it has seemed as though uniting these two realms implied either

ignoring fundamental design features of natural language, or ignoring physical/biological constraints. Unbounded recursion, and the apparent ease with which interrupted dependencies are dealt with do not seem to be susceptible to treatment within a single neurobiologically plausible architecture. We believe that our two-track architecture allows us to navigate through this dilemma and to make significant progress towards joining the cognitive and neural realms. Each property is natural within its own sub-component. The RR encoding provides a noise tolerant encoding of dominance and recursion, while the temporal encoding allows access to individual items. The two-track architecture is independently justified by the range of psycholinguistic data that it accounts for. Here we have demonstrated how it accounts for complexity phenomena. In Whitney and Weinberg (in preparation), we show that specification of reanalysis in terms of the temporal and RR encodings allows a more natural and comprehensive account of reanalysis cases than theories based on manipulations of a syntactic tree.

Of course, much work remains. Directions for future research include: (1) performing more experimental work to test predictions of the model; (2) extending the range of linguistic phenomena that the model handles, to include such topics as movement, government, and referent binding; (3) resolving further representational issues, such as how the processing rules are realized neurally, and how these rules are learned; (4) creating a model of sentence production, and specifying how the production and comprehension systems interact.

References

- Babyonyshev, M. & Gibson, E. (1999) The complexity of nested structures in Japanese. *Language*, 75, 423-450.
- Bach, E., Brown, C., & Marslen-Wilson, W. (1986) Crossed and nested dependencies in German and Dutch: A psycholinguistic study. *Language and Cognitive Processes*, 1, 249-262.
- Bastiaansen, M., van Berkum, J. & Hagoort P. (2002) Event-related theta power increases in the human EEG during online sentence processing. *Neuroscience Letters*, 323, 13-16.
- Berwick R. & Weinberg, A. (1984) *The Grammatical Basis of Linguistic Performance*, Cambridge, MA: MIT Press.
- Boland, J.E. & Tanenhaus, M.K. (1991) The role of lexical representations in sentence processing. In G.B. Simpson (Ed.) *Understanding Word and Sentence*. Amsterdam: North-Holland. 331-366.
- Chierchia, G. & McConnell-Ginet, S (1990) *Meaning and Grammar: An Introduction to Semantics*. Cambridge, MA: MIT Press.
- Christiansen, M.H. & Chater, N. (1999) Connectionist natural language processing: The state of the art. *Cognitive Science*, 23, 417-437.
- Elman, J.L. (1993) Learning and development in neural networks: The importance of starting small. *Cognition*, 48, 71-99.
- Frazier, L. (1985) Syntactic complexity. In D. Dowry, L. Karttunen, & A. Zwicky (Eds.) *Natural Language Processing: Psychological, Computational, and Theoretical perspectives*. Cambridge, UK: Cambridge University Press. 129-189.
- Gibson, E. & Thomas, J. (1997) Processing load judgements in English: Evidence for the Syntactic Prediction Locality Theory of syntactic complexity. Manuscript, MIT, Cambridge, MA.
- Gibson, E. (1998) Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68, 1-75.
- Gibson, E., & Thomas, J. (1999) Memory limitations and structural forgetting: The perception of complex ungrammatical sentences as grammatical. *Language and Cognitive Processes*, 14, 225-248.
- Gibson E. (2000). The dependency locality theory. In Marantz, Miyashita & O'Neil (Eds.), *Image, Language, Brain*. Cambridge, MA: MIT Press. 95-126.
- Gibson, E., Desmer, T., Watson, D., Grodner, D., & Ko, K. (submitted) Reading relative clauses in English.

- Hinton, G.E. (1990) Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46, 47-75.
- Hopf, J., Bader, M., Meng, M., & Bayer, J. (2003) Is human sentence parsing serial or parallel? Evidence from event-related brain potentials. *Cognitive Brain Research*, 15, 165-177.
- Hopfield, J.J. (1995) Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376, 33-36.
- Hummel, J.E., & Holyoak, K.J. (1997) Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*, 104, 427-466.
- Kanerva, P. (1995) A family of binary spatter codes. In F. Fogelman-Soulie and P. Gallineri (eds.), ICANN '95, *Proceedings International Conference on Artificial Neural Networks*, 1, 517-522.
- Klimesch W. (1999) EEG alpha and theta oscillations reflect cognitive and memory performance: a review and analysis. *Brain Research Reviews*, 29: 169-195.
- Lewis, R. L. (1996). Interference in short-term memory: The magical number two (or three) in sentence processing. *The Journal of Psycholinguistic Research*, 25, 93-115.
- Lewis, R. L., & Nakayama, M. (2001) Syntactic and positional similarity effects in the processing of Japanese embeddings. In M. Nakayama (Ed.) *Sentence Processing in East Asian Languages*. Stanford, CA: CLSI Publications. 85-110.
- Lisman, J.E., & Idiart, M.A.P. (1995) Storage of 7 +/- 2 short-term memories in oscillatory subcycles. *Science*, 267, 1512-1515.
- Marantz, A.P. (1984) *On the Nature of Grammatical Relations*. Cambridge, MA: MIT Press.
- Miller, G.A. & Chomsky, N, (1963) Finitary models of language users. In R.D. Luce, R.R. Bush, and E. Galantar (Eds.) *Handbook of Mathematical Psychology, volume 2*. New York: Wiley. 419-491.
- Pearlmutter, N. & Mendelsohn, A. (1998) Serial versus parallel sentence processing. Paper presented at the CUNY Conference on Human Sentence Processing.
- Plate, T.A. (1995), Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6, 623-641.
- Pollack, J. (1990). Recursive distributed representations. *Artificial Intelligence*, 46, 77-105.
- Raghavachari S., Kahana M., Rizzuto D., Caplan J., Kirschen M., Bourgeois B., Madsen J. & Lisman J. (2001) Gating of human theta oscillations by a working memory task. *Journal of Neuroscience*, 21:3175-3183.

- Rohde, D.L.T. (2002). A connectionist model of sentence comprehension and production. Unpublished PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.
- Seidenberg M.S. (1997) Language acquisition and use: Learning and applying probabilistic constraints. *Science*, 275, 1599-603.
- Shastri, L., & Ajjanagade, V. (1993) From simple associations to systematic reasoning. *Behavioral and Brain Sciences*, 16, 417-494.
- Shastri, L. (1999) Advances in SHRUTI – A neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence*. 11, 79-108.
- Shastri, L. (2001) A biological grounding of recruitment learning and vicinal algorithms. In S. Wernter, J. Austin, & D. Willshaw (Eds.) *Emergent Neural Computational Architectures Based on Neuroscience*. Heidelberg: Springer-Verlag. 348-367.
- Sheil, B.A. (1976) Observations on context-free parsing. *Statistical Methods in Linguistics*, 6, 71-109.
- Sussman, R.S. & Sedivy, J.C. (2001) The time-course of processing syntactic dependencies: Evidence from eye movements during spoken narratives. In J.S. Magnuson and K. M. Crosswhite (Eds.) *University of Rochester Working Papers in the Language Sciences*, 2, 52-70.
- Tiitinen, H., Sinkkonen, J., Rainikainen, K., Alho, K., Lavi-kainen, J., & Naatanen, R. (1993) Selective attention enhances the 40-hz response in humans. *Nature*, 364, 59-60.
- Warren, T. & Gibson, E. (2002a) The influence of referential processing on sentence complexity. *Cognition*, 85, 79-112.
- Warren, T. & Gibson, (2002b) Evidence for a constituent-based distance metric in distance-based complexity theories. Poster presented at the CUNY Conference on Human Sentence Processing.
- Whitney, C., & Berndt, R.S. (1999) A new model of letter string encoding: Simulating right neglect dyslexia. *Progress in Brain Research*, 121, 143-163.
- Whitney, C. (2001) How the brain encodes the order of letters in a printed word: The SERIOL model and selective literature review. *Psychonomic Bulletin & Review*, 8, 221-243.
- Whitney, C. (in preparation) Temporal representations in language processing. Doctoral dissertation. University of Maryland, College Park, MD.
- Whitney, C. & Weinberg, A. (in preparation) Reanalysis over lists, not trees.

Appendix

This section lists the sentences on which the RR-encoding simulation was tested. As discussed in the text, a sentence was specified as a sequence of two-character symbols. For ease of comprehension, we present an example sentence for each of the input sequences. The correct encoding was generated for all the sentences, except the RC/RC (12 and 13) and the RC/NC (29).

N1 V1 N2
1. The cat chased the rat.

Two clauses

N1 V1 C2 N2 V2 N3
2. The man knows that the cat chased the rat.

N1 R1 I2 V3 N3
3. The cat which was chased ate the fish.

N1 R1 N2 V2 V1 N3
4. The cat which the dog chased ate the fish.

N1 R1 V2 N2 V1 N3
5. The cat which chased the rat ate the fish.

N1 V1 N2 R1 V2 N3
6. The cat chased the rat which ate the cheese.

N1 V1 N2 R1 N3 V2
7. The cat chased the rat which the dog bit.

A1 N1 V1 N2 N3 V2 N4
8. After the cat chased the rat, the dog ate the meat.

N1 V1 N2 A1 N3 V2 N4
9. The dog ate the meat after the cat chased the rat.

C1 N1 V1 N2 V2 N3
10. That the dog ate the chocolate bothered Bill.

N4 C1 N1 V1 N2 V2 N3
11. The fact that the dog ate the chocolate bothered Bill.

Three clauses

- N1 R1 N2 R2 N3 V3 V2 V1 N5
12. The rat which the cat which the dog hates chased ate the cheese.
- N1 R1 N2 R2 V3 N3 V2 V1 N5
13. The rat which the cat which hates the dog chased ate the cheese.
- N1 R1 V2 N2 R2 N3 V3 V1 N5
14. The dog which chased the cat which the rat feared ate the meat.
- N1 R1 V2 N2 R2 V3 N3 V1 N5
15. The dog which chased the cat which chased the rat ate the meat.
- N1 V1 N2 R1 V2 N3 R2 V3 N4
16. The dog chased the cat which chased the rat which ate the cheese.
- N1 V1 N2 R1 V2 N3 R2 N4 V3
17. The dog chased the cat which chased the rat which the lion liked.
- N1 V1 N2 R1 N3 R2 V4 N4 V3
18. The dog chased the cat which the rat which ate the cheese feared.
- N1 V1 N2 R1 N3 R2 N4 V4 V3
19. The dog chased the cat which the rat which the lion liked feared.
- N1 R1 V2 C1 N3 V3 N4 V1 N5
20. The man who thinks that the cat chased the rat ate the cheese.
- N1 R1 N2 V2 V1 C3 N3 V3 N4
21. The man who the lion chased thinks that the rat ate the cheese.
- N1 V1 N2 R1 V2 C3 N3 V3 N4
22. The lion chased the man who thinks that the rat ate the cheese.
- N1 V1 C2 N2 V2 C3 N3 V3 N4
23. The man knows that the girl thinks that the cat chased the rat.
- N1 V1 C2 N2 V2 N3 A1 N4 V4 N5
24. The man thinks that the rat ate the cheese after the dog bit the cat.
- N1 V1 C1 A0 N2 V2 N3 N4 V4 N5
25. The man knows that when the cat chases the rat, the lion chases the dog.

N4 C1 N5 R1 N1 V1 V2 N2 V3 N3
26. The fact that the dog which Sue adopted ate the chocolate bothered Bill.

C1 N5 R1 N1 V1 V2 N2 V3 N3
27. That the dog which Sue adopted ate the chocolate bothered Bill.

N4 C1 N5 C2 N1 V1 N2 V2 N3 V3 N5
28. The idea that the fact that the dog ate the chocolate bothered Jane upset Bill.

N4 R1 N5 C1 N1 V1 N5 V2 V3 N3
29. The woman who the fact that the dog ate the chocolate bothered hit Bill.

four clauses

N1 R1 V2 C2 N2 V3 N3 A1 N4 V4 N5
30. The man who knows that the cat chased the rat after the dog ate the meat
V1 N6
ate the pie.