Using Relational Syntactic Constraints in Content-Addressable Memory Architectures for Sentence Parsing

Pedro Alcocer and Colin Phillips

University of Maryland
April 2012

Abstract

How linguistic representations in memory are searched and retrieved during sentence processing is an active question in psycholinguistics. Much work has independently suggested that retrieval operations require constant-time computations, are susceptible to interference, and operate under the constraint of a severely limited focus of attention. These features suggest a model for sentence processing that uses a content-addressable memory (CAM) architecture that accesses items in parallel while relying on only a limited amount of privileged, fast storage. A challenge for a CAM architecture comes from the potentially unbounded configurational relation c-command (equivalent to logical scope) that plays a pervasive role in linguistic phenomena. CAM is well-suited to retrieval of elements based on inherent properties of memory chunks, but relational notions such as c-command involve the properties of pairs of nodes in a structure, rather than inherent properties of individual chunks. To explore this problem, in this paper we adopt an explicit CAM-based model of sentence processing in the context of the ACT-R computational architecture (Lewis & Vasishth, 2005, *Cognitive Science, 29*, 375-419). We discuss why c-command is a challenge for CAM and explore algorithms for exploiting or approximating c-command online, and discuss their consequences for the model. We identify computational problems that any attention-limited, CAM-based model would have to address.

1. Introduction

Recent findings from psycholinguistic studies and computational modeling in sentence processing have raised interest in the use of cue-based retrieval mechanisms in the context of content-addressable memory architectures. These approaches assume that linguistic dependencies are formed using retrieval cues that target specific features of individual linguistic memory chunks. They make interesting predictions about profiles of interference errors in parsing (Lewis, Vasishth, & Van Dyke, 2006), and they can account for findings of rapid

retrieval times that are unaffected by the size of the linguistic dependency that is formed (McElree, Foraker, & Dyer, 2003; Martin & McElree, 2008). These approaches have also been tested using explicit computational models (Lewis & Vasishth, 2005). However, it is less clear how such models can exploit relational syntactic notions that play a key role in linguistic constraints, since these involve configurational properties of hierarchically structured representations (e.g., trees), rather than inherent properties of individual nodes in those trees. Here we focus on the c-command relation, exploring possible algorithms for encoding and exploiting this relation in a content-addressable memory architecture, and asking whether these algorithms allow the time cost of dependency formation to retain its size-independent character.

Linguistic representations must encode many different types of dependencies between different words and phrases, in order to capture such relations as subject-verb agreement, pronoun-antecedent relations, filler-gap dependencies, and verb phrase ellipsis. Some of these dependencies are very local, such as the relation between a verb and its object, or an adjective and the noun that it modifies. Other dependencies may span many words or clauses. Effective formation of linguistic dependencies requires mechanisms for encoding items in memory in a format that is suitable for subsequent use, and mechanisms for retrieving those items when they are needed to participate in a dependency. Retrieval mechanisms may be divided into serial mechanisms that take advantage of the graph structure in a sentence representation to search the representation node-by-node, and parallel mechanisms that take advantage of the information encoded in each individual node to examine entire representations at once. Serial algorithms can easily and accurately exploit relational constraints such as c-command, but they predict that the time to access a given item in memory should grow as the distance from the retrieval point to the target item increases. Conversely, parallel algorithms that query an entire memory representation

at once, in parallel, offer speed benefits, but at the potential cost of a loss of structural accuracy and the inability to exploit relational constraints.

Increasing numbers of experimental studies have examined the evidence for the different types of memory access mechanisms in sentence parsing. Some studies report evidence of the interference effects and length-invariant access times predicted by parallel access mechanisms (Badecker & Straub, 2002; McElree et al., 2003; Martin & McElree, 2008; Martin & McElree, 2009; Van Dyke & McElree, 2006; Vasishth, Brüssow, Lewis, & Drenhaus, 2008; Wagers, Lau, & Phillips, 2009). Other studies have found that at least some linguistic dependency formation shows the length-dependent and interference-free profiles that are expected if serial, structure-guided access mechanisms are used (e.g., Dillon et al., submitted; Dillon, Mishler, Sloggett, & Phillips, submitted; Phillips, Wagers, & Lau, 2011; Sturt, 2003).

In this paper, we attempt to assess the model of memory access that emerges when independently investigated properties of memory and sentence processing are considered together. These properties include constant-time retrieval, susceptibility to interference during retrieval, a limited focus of attention, and sensitivity to relational constraints such as c-command.

2. Three properties of memory access in sentence parsing

Three independently investigated properties of memory and retrieval have inspired recent interest in content addressable memory architectures (CAM) for sentence processing.

First, a number of findings suggest that retrieval times in linguistic dependency formation are constant, regardless of the length of the dependency. In a series of Speed-Accuracy Tradeoff (SAT) studies, McElree et al. (2003) found that as the amount of material increased between the

two elements of a linguistic dependency, measures of processing speed were unaffected. In an SAT task, participants are asked to make binary judgments about sentences at specific points in time. This allows for the examination of how sensitivity develops over time. When participants are forced to make a judgment immediately after a critical word appears, participants show very low sensitivity (measured in d' units) because too little time has elapsed for the relevant information to be processed. As more time becomes available, the likelihood of successful retrieval and processing of relevant information increases, and sensitivity increases correspondingly. In a variety of constructions involving filler-gap dependencies and long-distance subject-verb dependencies, McElree and colleagues found in a sensicality judgment task that interpolating varying amounts of sentence material between the two elements of a dependency impacted asymptotic sensitivity scores, but it did not impact the time course of the increase in sensitivity. They interpreted the effect of interpolated material on asymptotic accuracy as evidence for reduced probability of successful retrieval as the size of the representation increases (due to either decreased quality of the memory representation or to reduced distinctiveness of the retrieval cues). And they took the non-effect of interpolated material on SAT dynamics as evidence that retrieval is performed by a mechanism that accesses memory in parallel, examining all of the items in memory at once. This mechanism is situated in a content-addressable memory architecture that allows for direct access to memory representations without needing to traverse hierarchically intervening representations, as a serial retrieval mechanism might need to do. Similar findings of size-invariance have been reported for the resolution of verb phrase ellipsis (Martin & McElree, 2008; Martin & McElree, 2009).

Second, a number of studies have shown that retrieval is susceptible to interference, even from grammatically irrelevant material. Van Dyke and McElree (2006) had participants read

sentences while keeping in memory a list of words that overlapped in features with a critical word in the target sentence. For example, they measured reading times for the critical word, *sailed* and *fixed* in (1) while holding in memory a list of words that were related to *fixed,* but not *sailed,* e.g., *table, sink, truck*.

1. a. It was the boat that the guy who lived by the sea *sailed* in two sunny days.
   b. It was the boat that the guy who lived by the sea *fixed* in two sunny days.

Van Dyke and McElree found that in this memory load condition participants were slower to read the critical word when it overlapped semantically with the words in the memory list, i.e., *fixed* in (1b) was read more slowly than *sailed* in (1a). There was no corresponding slowdown in a condition with no concurrent memory task. Van Dyke and McElree attributed the effect of overlapping features on reading times to the process that occurs when the critical verb must retrieve its clefted object (*boat* in (1-2)). Although the memory list should be irrelevant to the task of resolving the filler-gap dependency, the overlap effect suggests that it was not irrelevant to the retrieval process. The overlapping words could lead to slower reading times in at least two ways. They could make successful retrieval of the filler phrase slower by reducing the distinctiveness of the memory representation of the filler. Or they could increase the likelihood of incorrectly retrieving a word from the memory list instead of the filler. Either account of the slowdowns is consistent with a content-addressable memory architecture, and is more surprising in a serial, structure-guided memory architecture..

Another set of findings that implicate CAM are also often described in terms of retrieval interference, but they involve a rather different phenomenon. A number of studies have shown

that the processing of ungrammatical sentences is actually facilitated by the presence of a structurally-irrelevant but feature-appropriate element. For example, Wagers and colleagues (2009) tested the processing of subject-verb agreement in sentences like (2a-d), manipulating the grammaticality of adjacent subject-verb pairs (2ab vs. 2cd) and the match or mismatch between the critical verb (*wave(s)*) and the head of the relative clause (*runner(s)*) (2ad vs. 2bc). The relative clause head is referred to as the *interfering* noun, as it is grammatically irrelevant to resolution of the subject-verb agreement relation, but in a CAM architecture it might exhibit a partial match to the retrieval cues at the verb. They found, unsurprisingly, that reading times on the verb were generally slower in sentences with ungrammatical agreement. But the effect of ungrammaticality was sharply reduced in the ungrammatical condition where the interfering noun matched the plural agreement on the verb (2d). Also, in speeded acceptability judgment tasks (2d) was rated as acceptable substantially more often than (2c). This effect has been described as an *illusion of grammaticality* (Phillips et al., 2011). No corresponding *illusion of ungrammaticality* effect was found in (2ab). Other studies have found similar facilitatory interference effects in the processing of agreement (Lago, Alcocer & Phillips, 2011; Pearlmutter, Garnsey, & Bock, 1999; Staub, 2009).

2. a. The runner who the driver *waves* to every morning always says hi.

   b. The runners who the driver *waves* to every morning always says hi.

   c. *The runner who the driver *wave* to every morning always says hi.

   d. *The runners who the driver *wave* to every morning always says hi.

The facilitatory effect of grammatically irrelevant items on agreement processing is expected in a noisy memory architecture in which retrieval cues are simultaneously compared with all items in memory, leading to the possibility of misretrieval of an incorrect agreement controller in some proportion of trials. Note that this does not imply that the retrieval cues ignore structural constraints, only that the those constraints do not block structurally inappropriate material from participating in the retrieval process. The lack of interference effects in the grammatical conditions can be explained by the fact that in the grammatical conditions the correct subject perfectly matches the structural and morphological retrieval cues, and hence the likelihood of misretrieval of another noun is reduced. Unlike the slowdown effects reported in the Van Dyke & McElree (2006) memory-load manipulation, the facilitatory interference effects observed in agreement processing more directly implicate misretrieval, as they are not easily explained in terms of reduced quality of the encoding of the items in memory.

Interference effects in a number of other linguistic dependencies have also been analyzed in terms of misretrieval in a CAM architecture, but the interpretation of those findings is less clear cut. The processing of the *negative polarity item* (NPI) *ever* shows a robust illusion of grammaticality effect. This has been interpreted as a case of misretrieval (Vasishth et al., 2008), but others have argued that it instead reflects over-application of pragmatic licensing mechanisms (Xiang, Dillon, & Phillips, 2009). In the processing of pronouns (e.g., *she, him, we, them*) or reflexives (e.g., *himself, themselves*) many studies have tested whether the retrieval of antecedents is constrained by linguistic binding constraints (Chomsky, 1981; Büring, 2005), such that only clause-mate antecedents are considered for reflexives and clause-mate nouns are excluded for pronouns. Some studies report effects of interference from structurally inappropriate nouns (Badecker & Straub, 2002; Patil, Vasishth, & Lewis, submitted), and others

report that interference effects are either absent or delayed (Clifton, Frazier, & Deevy, 1999; Clackson, Felser, & Clahsen, 2011; Dillon et al., submitted; Nicol & Swinney, 1989; Sturt, 2003; Xiang et al., 2009). However, none of these studies have shown a facilitatory interference effect like the one that is consistently observed in agreement processing.

Third, sentence processing must operate under the constraint of a severely limited focus of attention, i.e., the number of items that are immediately available for processing without additional retrieval operations is quite small. This is to be contrasted with the capacity of long-term memory, which is very large. Considering a wide variety of data, Cowan (2001) proposes that the number of items that can be actively held in working memory is only around four. McElree (2006) argues that the focus of attention may be as small as just one item. This claim is based on SAT dynamics profiles found in a number of studies in which all items exhibit similar dynamics, except for one item that is accessible substantially faster than all others.

If updating of sentence representations is possible only for items in the focus of attention, and if the focus of attention is as limited as current evidence suggests, then this implies that sentence parsing requires constant shunting of information in and out of the focus of attention, given the number of different items that participate in different linguistic relations in any sentence of even moderate complexity.

Taken together, these three properties of memory mechanisms constrain the design of a psychologically plausible sentence processing model. The mechanism that underlies retrieval must be devised in such a way that these three properties emerge from it. They suggest a model for sentence processing that uses a content-addressable memory architecture that accesses items in parallel while relying on only a very limited amount of privileged, fast storage. The exact

mechanism by which parsing takes place in such a model is left open, though it is certainly constrained by the properties of the memory architecture.

3. The ACT-R Model of Sentence Parsing

The three properties described above place specific constraints on a model of human sentence parsing.

First, a parallel retrieval mechanism that shows the property of constant-time access, must, in addition to retrieving items from memory in constant time, also be able to update memory encodings in constant time as each new word arrives. Given that all of working memory is examined simultaneously, the number of computational steps (and hence time) required to encode or retrieve a representation in memory should not scale with the overall size of the stored representation. In order to maintain the observed constant-time property, a parallel system must be parallel "all the way down". It should never resort to using a serial algorithm for encoding of information that must be updated prior to retrieval. A parsing cycle should take the same amount of time no matter the distance between two parts of a dependency. This contrasts with a serial algorithm that traverses the tree, node by node, and for that reason should take longer to retrieve an element that is more distant from the site that initiates retrieval.

Second, a suitable model should use a mix of structural and non-structural cues in its retrieval operations, in order to capture the finding that retrieval operations do show sensitivity to structural constraints on linguistic dependencies, but that these constraints can be overridden by structurally inappropriate targets that better match the non-structural retrieval cues. Examples of structural cues might include *subject*, or *main clause*, and examples of non-structural cues might

include such features as *plural, masculine, animate*, or *quantificational*. A suitable model should also be able to capture in terms of these features all of the structural notions that are known to be relevant in on-line parsing. We return below to the challenge of encoding relational notions in terms of retrieval cues.

Third, a suitable model should take seriously the findings about the severely limited focus of attention in human parsing. A model that presupposes the ability to simultaneously process large amounts of information is unlikely to be psychologically plausible.

One attempt at specifying such a model while being explicit about the details of parsing is described by Lewis and Vasishth (2005) and Lewis et al. (2006) (henceforth L&V). These authors propose a detailed model of the memory operations that subserve parsing, instantiated in the ACT-R cognitive framework (Anderson, 2005) and adopting the cognitive processing principles that the framework entails. By adopting these principles the L&V model is able to capture the three key properties of memory in parsing that we have outlined.

The first principle adopted from the ACT-R framework is that memory is stored in "chunks" consisting of named sets of feature-value pairs. Chunks in this model correspond to nodes in a syntactic tree. Chunks are stored unordered in a content-addressable working memory (CAM), which is accessed in parallel. The parallel access property captures the size-invariant property of retrieval times.

Second, because of the extremely limited focus of attention, all work is done in a series of buffers that only have the capacity to contain a single item. L&V specify four buffers in their model: a buffer which contains information relevant to the current syntactic category goal; a buffer which contains the current problem state; a retrieval buffer, which holds a chunk retrieved from working memory; and a lexical buffer, which holds an item retrieved from the lexicon.

These four buffers do not map directly onto quantitative claims about the scope of focal attention, but they certainly honor the notion that sentence processing requires constant shunting of information in and out of limited capacity buffers.

Third, memory chunks can have varying levels of activation. The overall activation of a chunk is computed according to Equation 1, which is a sum of the chunk's baseline activation (B_i), the additional activation it receives as a function of cue-match with the search query ($\sum_j W_j S_{ji}$), mismatch penalties ($\sum_k PM_{ki}$), and noise (ε). This additional activation is a function of W_j, the weights of the cues in the search query, and S_{ji}, the strength of association between cues j and chunk i. The latter value is computed using Equation 2. Where S is the maximum associative strength and fan_j is the "fan" parameter, which is a function of the total number of items associated with the cue j. So, the more often a cue appears in a sentence representation, the weaker is the associative strength between that cue and any individual memory chunk. Mismatch penalties are a function of P, a typically negative-valued parameter that determines how much activation is lost due to cue mismatch, and M_{ji}, a logical value that is 1 if cue j does not match with chunk i and 0 otherwise. Finally, ε is a noise parameter that models the stochasticity of real processing.

$$A_i = B_i + \sum_j W_j S_{ji} + \sum_k PM_{ji} + \varepsilon$$

Equation 1.

$$S_{ji} = S - \log(fan_j)$$

Equation 2.

The activation function is used in a winner-takes-all scheme to select which chunk will be retrieved by any set of retrieval cues. The chunk with the highest activation at the time that memory is probed is the one that is retrieved. Since retrieval does not require perfect matches to retrieval cues, since activation values include a noise component, and since the activation function does not single out structural features as privileged, the retrieval mechanism is able to capture the appearance of interference effects. Inhibitory interference effects of the kind described by Van Dyke and McElree can be captured in terms of the fan effect or in terms of misretrieval. Facilitatory interference effects like those found in agreement attraction can be captured in terms of misretrieval of memory chunks that partially match the retrieval cues. See Dillon (2011) for discussion of the interpretation of different kinds of interference effects.

Finally, parsing rules in the L&V model are stored as rules in procedural memory. Rules take the form of conditions, which depend on the current state of the buffers, and actions, which are performed if the conditions are met.

Chunks gain their features through a series of interactions with the lexicon, working memory, and the various buffers mediated by parsing rules. As a new word is input, the lexicon is accessed and the new word's lexical entry is placed into the lexical buffer. Semantic features (e.g., person, number) and lexical category are read from the lexical entry. Next, to create a syntactic relation to another chunk in memory, retrieval cues are derived from the item in the lexical buffer and the goal state of the parser. A search is executed based on those retrieval cues, causing a chunk to be retrieved from memory. Finally, a new chunk is created in working memory based on the content of the lexical buffer, the retrieval buffer, and the state of the goal buffer. The goal buffer is then updated and the parsing cycle begins again when the next word is input.

The parallel, simultaneous nature of retrieval is what enables this model to perform retrievals quickly. The use of buffers that can only hold one chunk is what limits the focal attention of the model. The use of chunks, which can potentially overlap in features, and the activation equation that similarly activates chunks with similar features, jointly give rise to the susceptibility of retrieval to interference from structurally inappropriate chunks in memory. Though these chunks mismatch the structural component of the retrieval cues, they sufficiently match in features with the other retrieval cues that they sometimes are not distinguishable from chunks in structurally appropriate positions. This can lead to misretrieval effects, the behavioral consequences of which are (facilitatory) interference effects.

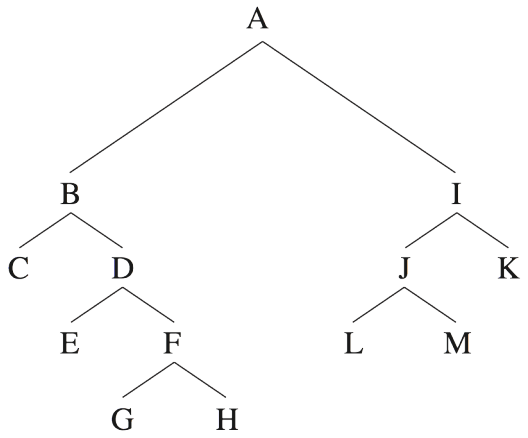
4. A fourth property: relational structural constraints

The CAM-based model that L&V propose is compatible with the three properties of memory in sentence processing that we have highlighted. A fourth property, that is strongly motivated by the nature of linguistic constraints, is less obviously compatible with a CAM-based model, though it is easier to capture in a model with a serial search mechanism. Many linguistic dependencies are subject to the requirement that the elements in the dependency stand in a c-command relation to one another. This is a relational property that obtains between pairs of nodes in a hierarchical structure, contrasting with structural properties that may be regarded as inherent properties of the encoding of an individual memory chunk, such as *subject* or *topic* or *item in clause N*, where *N* is the index of a specific clause.

Formally, a node *P* c-commands another node *Q* iff *P* does not dominate *Q*, and the first branching node that dominates *P* also dominates *Q* (Reinhart, 1983). Another way to think of the

c-command relation is as a relation that a node has with its sister and all of its sister's descendants. In (3) there are many c-command relations. For example, node B c-commands nodes I, J, K, L, and M; node C c-commands nodes D, E, F, G, and H; node L c-commands node M. Node C does not c-command node I or any of its descendants because B dominates C but does not dominate I. Node B does not c-command C, because B dominates C.

(3)



The c-command relation is pervasive in descriptions of linguistic dependencies. For example, bound variable readings of pronouns require that the pronoun be c-commanded by a suitable antecedent. The pronoun *he* in (4) can only take a negatively quantified noun phrase as its antecedent via a bound variable interpretation. This is possible in (4a), where the main clause subject *no student* c-commands the pronoun, but not in (4b) where the noun phrase *no physics professor* is embedded inside a relative clause, where it fails to c-command the pronoun. Sentence (4b) is, of course, an acceptable string in English, but not under the interpretation indicated in (4b). Importantly, the antecedent for a bound variable pronoun can be arbitrarily far from the pronoun, as long as it c-commands the pronoun.

4. a. No student$_i$ [that listened to this physics professor] thinks that he$_i$ is a genius.

b. * The student [that listened to no physics professor$_i$] thinks that he$_i$ is a genius.

C-command is required for other relations, including filler-gap dependencies (the filler must c-command the gap), reflexive binding (the antecedent must be a c-commanding clausemate of the reflexive), control (the controller must c-command the controlled argument position), to name but a few. The licensing of negative polarity items (NPIs) like *any* or *ever* is another phenomenon that is often described in terms of c-command. In (5a-b) the NPI *ever* is licensed by the presence of c-commanding negation, which may appear in the same clause (5a) as the NPI or another clause (5b). The absence of the negation makes the NPI unacceptable (5c). Importantly, when the negative expression is embedded inside a relative clause and fails to c-command *ever* (5d), NPI licensing also fails. It is controversial whether NPI licensing genuinely involves an item-to-item dependency along the lines of binding and agreement. Classic and recent accounts of NPI licensing analyze the c-command requirement as a consequence of semantic/pragmatic licensing requirements (Ladusaw, 1979; Kadmon & Landman, 1996; Chierchia, 2006). But others have argued that at least some kinds of NPI licensing does genuinely involve an item-to-item dependency (Giannakidou, 2011). The case of NPI licensing is relevant to the current discussion, as c-command has been invoked as a potential retrieval cue in a CAM architecture, in order to capture the finding that fleeting illusions of grammaticality occur in sentences like (5d) (Vasishth et al., 2008).

5. a. No student could ever answer the final question on the exam.

b. No student thought that his teacher could ever answer the final question.

c. *The student could ever answer the final question on the exam.

d. * The student [that listened carefully to no teacher] could ever answer the final question on the exam.

The fact that c-command is a pervasive constraint on linguistic dependencies, as defined by off-line judgments, is no guarantee that it plays a similarly pervasive role in on-line retrieval processes. But there is evidence that it does impact retrieval. C-command could impact on-line processes in multiple ways. First, it could serve a gating function, such that retrieval operations could target only those structural positions that c-command the retrieval trigger, excluding all other positions from consideration. Second, it could function as a non-exclusive constraint on retrieval, such that c-commanding items count as better matches to the retrieval cues, but other items are nevertheless considered as possible targets. Third, c-command could act as a constraint on on-line dependency formation without necessarily implicating retrieval operations *per se*.

Many on-line studies probe the formation of linguistic dependencies that are subject to a c-command restriction. But this is not sufficient to demonstrate the on-line effects of c-command constraints. That requires a demonstration that on-line processes distinguish c-commanding and non c-commanding pairs of elements, something that has been tested in a narrower range of studies.

There is good evidence that c-command constrains on-line dependency formation. In multiple studies in English, Japanese, and Russian, our group has found that the search for pronoun antecedents in backwards anaphora contexts is impacted by a c-command constraint. A pronoun may precede its antecedent (6ac), but it may not c-command its antecedent (6bd). This constraint is known as *Binding Principle C* (Chomsky, 1981), and it appears to exclude

c-commanding pronoun-name pairs from consideration in the search for antecedents (Aoshima, Yoshida, & Phillips, 2009; Kazanina, Lau, Yoshida, Lieberman, & Phillips, 2007; Kazanina & Phillips, 2010). These findings show that the parser must be keeping track of c-command relations on-line, but they do not necessarily implicate c-command specifically in retrieval processes, as they involve a forwards search from a pronoun to a potential antecedent.

6. a. Her$_i$ mother thinks that Mary$_i$ is a genius.

b. * She$_i$ thinks that Mary$_i$ is a genius.

c. While she$_i$ was in Paris, Mary$_i$ read three novels.

d. * She$_i$ read three novels while Mary$_i$ was in Paris.

A number of studies that more directly implicate retrieval processes show that c-commanding elements are favored over non c-commanding elements. For example, processing of reflexive pronouns shows greater sensitivity to the match between the reflexive and a c-commanding subject NP than to the match with other non c-commanding NPs (Dillon et al., submitted; Nicol & Swinney, 1989; Patil et al., submitted; Sturt, 2003). This bias is clear, despite disagreement on the question of whether or not the bias is absolute. However, these findings do not directly implicate c-command in retrieval, despite the fact that reflexives must be c-commanded by their antecedents. This is because the antecedents in those studies were consistently a clause-mate subject NP, and hence the retrieval could have simply used *subject of the same clause* as a structural cue.

Tests of the sensitivity of retrieval mechanisms to c-command constraints require measures of the formation of longer-distance relations that are not so easily recast in terms of

alternative structural notions such as *subject-of-current-clause*. The evidence on this issue is limited at present, but a number of findings suggest more general sensitivity to c-command in retrieval. In studies of bound variable pronoun processing, comprehenders show sensitivity to the (un)availability of a c-commanding antecedent (Kush, Lidz, & Phillips, 2012). In studies of the processing of control in English adjunct clauses (Parker, Lago, & Phillips, 2012) and of null subjects in Brazilian Portuguese (Alcocer, França, Maia, & Phillips, 2010), comprehenders show success in retrieving an antecedent that is the subject of the next higher clause (c-commanding), rather than a non c-commanding subject. Studies of NPI licensing show comprehenders are sometimes subject to illusory licensing from non c-commanding negation, but there are demonstrations of contexts where only c-commanding negation is considered (Parker & Phillips, 2011), and even in the cases where illusory licensing occurs the c-commanding licenser generally has a more powerful effect than a spurious non c-commanding licenser (Vasishth et al., 2008; Xiang, Dillon, & Phillips, 2006). Recall that although NPI licensing is descriptively subject to a c-command constraint, it is controversial whether the c-command requirement is directly stated in the licensing requirements or whether it is a consequence of independent semantic/pragmatic licensing requirements.

Notwithstanding the need for further evidence on the on-line impact of c-command constraints, it is safe to say that c-command has powerful effects on linguistic dependencies in general, and that there is no clear evidence that it is simply ignored in on-line retrieval processes. We therefore assume for the remainder of this discussion that it is a desirable property of a parser to be able to utilize c-command as a structural constraint on retrieval processes. This motivates the need to discover a mechanism for computing c-command in a system as computationally constrained as the L&V model.

How might the c-command relation be encoded in a cue-based, parallel system like the L&V model? A tempting solution to the problem is to simply make c-command a binary (or, similarly, privative) feature, as Vasishth and colleagues suggest in their study of illusory NPI licensing (Vasishth et al., 2008: p. 695, Fig. 2). Under this approach, a hierarchically higher node is tagged [c-commander: +]. When a lower node is accessed that requires that a c-commanding negative element be retrieved, [c-commander: +] is added to the retrieval cues. Search then returns the appropriate item because it is tagged [c-commander: +].

We regard the privative feature approach to be a non-starter, because the approach incorrectly treats c-command as a fact about a single node, whereas c-command is a two-place configurational relation between pairs of nodes – a *c-commanding* node and a *c-commanded* node. Simply encoding the fact that a given node c-commands some other node is not useful in retrieval processes, because what is relevant is whether a node specifically c-commands the node that is triggering the current retrieval process. In fact, the implemented model that Vasishth and colleagues (2008) use to model illusory NPI licensing does not use the *c-command* feature that is shown in their illustration of the model. They instead distinguish the structurally appropriate and inappropriate licensers using structural cues such as *subject* and *nominative* that are readily encoded as features of individual nodes in a CAM architecture. We must therefore look elsewhere for more realistic approaches to using c-command in a CAM architecture.

5. Approaches to encoding c-command

We now consider a series of approaches to encoding c-command within a CAM-based architecture. The approaches vary in terms of how fully they capture the c-command available to

a serial processor that can exploit the graph structure of a tree in its search processes. First, we consider an approach that encodes structure as content, tracking c-command information through the parse and encoding it explicitly on each chunk. This approach captures the full range of c-command relations, though at some cost to computation and representation. Next, we consider two approaches that allow for simpler encodings, but at the expense of capturing a smaller subset of c-command relations. Finally, we briefly consider two options that address the problem by assuming that retrieval processes are not able to effectively use c-command constraints. In each case we consider how the encoding of c-command might impact predictions for findings such as the size-invariant timing of retrieval.

5.1 Full encoding: C-command as a relational feature ('structure-as-content')

HOW IT WORKS. The first approach aims to provide a comprehensive encoding of c-command relations via an algorithm that annotates memory chunks with information about non-adjacent portions of the tree. This additional information makes it possible to narrow down the search space during retrieval to only eligible c-commanders. This approach encodes the two-place c-command relation by including on each memory chunk a feature-value pair for every other chunk that c-commands it. To retrieve eligible c-commanders of the node currently in focal attention, the c-commander list for that node is added to the set of retrieval cues.

For expository purposes, we first describe an algorithm for annotating chunks with c-command information as it would be applied top-down and offline if the full parse of the sentence were available at the outset.

C-COMMAND-AS-CONTENT, OFF-LINE VERSION. Starting with the root node:

1. Retrieve the current node N's parent node P and copy the list of P's c-commanders onto N, if it has any.

2. Verify whether P has a child other than N. If it does, this is N's sister node S.

3. Add S's label to N's list of c-commanders.

4. Proceed to the next node (top-down) and repeat until all the nodes are exhausted.

This algorithm encodes on each node in a sentence a list of all of the other nodes on the tree that c-command it. Because the furthest node it needs to examine during encoding is at most two nodes away (i.e., the path from the current node to its sister), it has the benefit of being a local operation and therefore requiring a constant amount of time to update the encoding at each node.

However, because parsing is incremental and hierarchical, material that is linearly adjacent in a sentence is not always hierarchically adjacent in a tree that corresponds to that sentence. During parsing, non-terminal structure must be projected for incoming words to attach to. To be useful for parsing, this algorithm would also need to work from left to right on a developing representation, taking into account structure that is projected due to syntactic expectations. The online version of this algorithm is similar to the offline version, except that it requires a processing queue to temporary hold projected structure. The processing queue requirement introduces some challenges that we address below.

C-COMMAND-AS-CONTENT, ON-LINE VERSION Starting at the beginning of a sentence:

1. Retrieve the current node N's parent node P and copy the list of P's c-commanders onto N, if it has any.

2. Verify whether P has a child other than N. If it does, this is N's sister node S.

3. Add S's label to N's list of c-commanders.

4. Project structure based on N and the state of the goal buffer.

5. Add nodes in projected structure to a processing queue.

6. Apply steps 1-4 to each of the nodes in the processing queue.

7. Receive the next word and repeat steps 1-6.

With information encoded on each node about its list of c-commanders, it becomes possible to issue retrieval cues that make reference to this information. When a new word is added to the parse, the c-command list must be updated on each new node before any retrieval operations can be initiated. When the current node requires that a c-commanding node be retrieved in order to satisfy some linguistic constraint, the "c-commanded-by" features are read off the current node and added to the list of retrieval cues. The feature values, in this case, are node IDs. During the retrieval process, the c-commanding chunks receive additional activation, effectively constraining the chunks under consideration to the c-commanders.

A variant of this approach would store the lists of c-command relations in a single table for the entire sentence. As far as we can tell, this alternative would have similar consequences to the version that we describe here.

MODEL CHANGES AND EMPIRICAL CONSEQUENCES. This approach would require changes to the L&V model with respect to the limitations on the size and number of memory buffers. The

listing of all c-commanders of a node as part of the content of each node amounts to a proliferation of structural information that may be cumbersome. In addition, the use of lists of potential c-commanders as retrieval cues probably requires a modification of the calculation of cue matching.

The problem of projecting structure is a general parsing problem and not unique to the model we are considering. It is an obstacle, however, that projected nodes must be temporarily kept in a queue for additional processing beyond what parsing typically calls for (viz., computing c-command relations vs. mere attachment). Because the amount of projected structure is, in principle, arbitrarily large, it may unreasonably stretch the notion of a limited focus of attention and limited buffer sizes. In practice, however, the limited depth of left branches in normally occurring language might neutralize this concern.

This approach to enforcing c-command constraints requires a different approach to retrieval cue matching than is assumed in the L&V model. In the existing version of the model, the best match to a set of retrieval cues is a memory chunk that perfectly matches all of the cues. In this way, chunks in memory that perfectly match with the retrieval cues receive large boosts in activation, while those that mismatch in at least some features are penalized, providing the correct target with a much greater probability of being successfully retrieved. In contrast, the structure-as-content approach to c-command requires that the retrieved item match just one of the node IDs from the list of c-commanders of the current node. If this list is used in an unmodified version of the L&V model, then this should predict that even a perfectly matching target item should elicit a substantial mismatch penalty, due to the fact that a given node can only ever match one of the nodes on the list of c-commanders. Similarly, in any sentence of moderate complexity, the number of nodes that partially match the list of c-commanders should mean that

the fan parameter (Equation 2 above) should predict a low association strength between the target and the current node.

One might, in principle, devise a novel version of the activation function in Equations 1 and 2 above that treats the match to items in the c-commander list differently than other cases of cue matching, such that there is no fan effect and no mismatch penalty in the case where a memory chunk mismatches all but one of the node IDs on the c-commander list. However, it remains unclear how to modify the activation function to achieve this result, and whether this would amount to giving privileged status to structural cues, thereby dropping one key feature of the L&V model.

The fact that c-command is treated as a retrieval cue on a par with other content cues should over-generate predictions of spurious retrieval. Consider a case where licensing of an incoming word requires retrieval of a specific type of c-commanding element, such as a quantificational NP for a bound variable pronoun, or a negative operator in the case of NPI licensing. Previous studies have focused on the question of whether semantically compatible non c-commanding elements are occasionally mis-analyzed as licensors (Dillon et al., 2009; Vasishth et al., 2008; Xiang et al., 2009), and conclusions are mixed. But the structure-as-content approach to c-command predicts that many semantically incompatible c-commanding elements should also be mis-analyzed as licensors. For example, in a sentence like (7a) the NPI *ever* is clearly unacceptable. But there are many chunks in memory that c-command *ever*, and hence we might expect many instances of illusory licensing based on partial matches to the retrieval cues for *ever*. In fact, these instances of illusory licensing should be just as common as illusory licensing based on partial matches to non-structural cues. For example, the complementizer *that*

in (7a) would count as an equally good partial match to the retrieval cues of *ever* as does the non c-commanding negative NP in (7b).

7. a. *John said that Mary's sister will ever finish her novel.

   b. * The student [that listened carefully to no teacher] could ever answer the final question on the exam.

We know of no evidence that partial matches based on structural cues lead to illusory licensing in the same way that partial matches based on non-structural cues sometimes do. This suggests that an alternative approach is needed that treats the matching of structural retrieval cues differently than the matching of non-structural retrieval cues.

This approach also introduces a great deal of undesirable redundancy to the sentence representation in working memory. Under this approach, every node holds a substantial amount of information about the overall structure of the representation. This, along with the requirement that node IDs be cues, causes this model to be rather powerful.

5.2 Limited encoding I: Dynamic [command-path] feature

HOW IT WORKS. An alternative to an encoding scheme that captures all c-command relations in an unfolding structure is an encoding scheme that provides rapid CAM-based access to information about a subset of the c-command relations in a structure at any intermediate parse state. We describe two such options here. Their attraction is that they provide an encoding that is more straightforwardly consistent with L&V's implementation of a CAM architecture. Their disadvantage is that they provide incomplete c-command information.

The first of the limited encoding options is guided by the intuition that at any moment in time only a subset of the c-command relations in a structure is relevant to the parser's current goals.[1] In particular, the parser may have the greatest need for information about the elements that c-command the word or phrase that it is currently attaching into the structure. By focusing on just those elements, it may be possible to simplify the encoding of c-command. The specific suggestion is that each memory chunk in a sentence representation has a single [command path] feature, which has binary on/off values. A chunk's command-path feature is set to ON when it is a c-commander of the current input node, and it is set to OFF when it is not a c-commander of the current input node.

LIMITED ENCODING I: DYNAMIC COMMAND-PATH FEATURE

Starting at the beginning of the sentence:

1. Set the command-path (CP) feature of the current input node to OFF
2. If the current node has a parent, set its CP feature to OFF
3. If the parent node has a sister, set its CP feature to ON
4. If the parent has a parent of its own, repeat steps 2-3
5. If the current node has a sister, set its CP feature to ON
6. For any node whose CP feature is turned ON, set the CP feature of all nodes that it dominates to OFF.
7. Repeat this procedure for the next input node.

An important feature of this approach is that the command-path feature of a node is not fixed after it is initially set. Rather, it is dynamically updated when relevant attachment decisions

[1] We thank Dave Kush and Titus van der Malsburg for suggesting this encoding approach to us.

are made. The main changes occur when a node's c-command value is set to ON, by virtue of its sister being the current input node or a dominator of the current input node. Turning on the command-path value for that node has the consequence that all nodes that it dominates are set to OFF. Once a node's command-path is switched from ON to OFF, it is unlikely that it will be turned back on at a later point. In practice, this procedure has the consequence that command-path feature values change when the parser 'pops out' of a left branch in a parse. The deeper the left branch that it pops out of, the greater the number of nodes whose command-path feature needs to be updated.

When a new element is enters the parse that forms a dependency with a c-commanding element the parser must first update the encoding of the command-path features in the current parse tree. Once that step is complete the newly introduced element can initiate retrieval of a suitable memory chunk that it can form a dependency with, including [command-path: ON] among the list of retrieval cues. This retrieval cue can be used with the activation functions in Equation 1 and 2 above.

MODEL CHANGES AND EMPIRICAL CONSEQUENCES. The main attraction of this encoding is that it allows retrieval of c-commanders of the current input node, by virtue of a single feature on all memory chunks that can easily be exploited in L&V's CAM-based architecture.

The main change to the L&V model that this encoding of c-command would require involves the dynamic updating of the command-path feature of nodes in the structure at each point that a new node is introduced. In most cases, when the parser is elaborating a right-branching structure, very few nodes in the existing parse should be affected by the introduction of the new node. Only when a high attachment is made, effectively popping the parser out of a left branch, must more nodes undergo updating of their command-path feature. The deeper the

left branch, the more nodes much be updated. This effect of attachment site on the amount of updating may have positive empirical consequences: it is known that increased parsing difficulty is associated with popping out of a left branch, so that is consistent with the predictions of this approach.

A related consequence, which might not be so positive, is that the updating of the command-path features upon introduction of a new node presupposes that a suitable attachment site for the new node has already been chosen. This means that attachment decisions presumably cannot make reference to c-command relations, for the reason that those relations are not available until after the attachment site has been determined.

The most important limitation of this encoding scheme is that it only provides an encoding of the nodes that c-command the current input node. This should be sufficient for many cases of linguistic dependency formation that are subject to c-command constraints, since those constraints typically have to be satisfied as soon as the dependent element is introduced into the parse. However, there are cases where this encoding falls short. First, in situations where the current input node needs to c-command an earlier node in the sentence, i.e., the opposite of what the proposed encoding tracks. Cases where the current node needs to c-command previously encountered material can be found in head-final languages like Japanese, where heads of relative clauses appear to the right of the gaps that they are associated with, and where negative polarity licensers often can appear to the right of the NPIs that they c-command. Such cases are common across languages.

Second, in situations where a c-command relation is not apparent at the point when a word is introduced but becomes necessary at a later point, this encoding scheme would again fail, as it is limited to only encoding the c-commanders of the current input node. Such situations

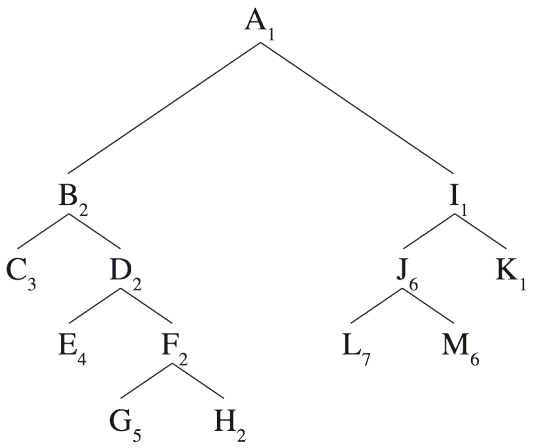
can arise when the introduction of a new, phonologically overt node signals the need for a c-commanding licenser for an earlier node. We have examined just such a scenario in Brazilian Portuguese (Alcocer et al., 2010). Brazilian Portuguese is a partial null subject language. Unlike Spanish, where null subjects can be freely introduced and can refer to any suitable discourse element, Brazilian Portuguese null subjects in embedded clauses are possible only when they are licensed by a c-commanding subject of a higher clause. But the presence of the null subject becomes apparent only when subsequent overt material is introduced. When an inflected verb is introduced following a complementizer, this is a cue that a new clause must be created with a finite verb and a null subject. In the current encoding scheme the current input node would be the finite verb, and the encoding of the tree would reflect its command path, but at that moment the command-path of the null subject would be most relevant to licensing the null subject. In this particular case it would be sufficient for the command path of the verb to be used as a proxy for the command path of the null subject. Further work is needed to determine the range of situations in which the parser is called upon to access information about c-commanders of nodes other than the current input.

5.3 Limited encoding II: Dominance spines as heuristic

HOW IT WORKS. Another alternative to fully encoding c-command is to use an easier-to-compute heuristic that does not require the parser to maintain an ongoing record of all possible c-command relations in the sentence. One possible heuristic is based on what we refer to as *dominance spines*. This heuristic captures many c-command relations, but not all of them. A dominance spine can be defined as a chain of dominance relations that proceeds down the rightward branches of a tree. Every time that a branching node branches leftward, a new

dominance spine is created. An example of dominance spines can be seen in the tree structure in (8). Nodes A, I, and K are a sequence of nodes in which each successive node is the right-hand daughter of the previous one. For this reason they are subscripted with the same dominance spine index. Similarly, nodes B, D, F, and H also form a dominance spine. In contrast, nodes C, E, and F each have a unique spine index, as they are left branches that have no daughters.

(8)



The attractions of dominance spines for current purposes are: (i) they assign to each node a single index, which is easily encoded as a feature-attribute pair that can be treated as part of the content of the node, and hence is suitable for standard retrieval mechanisms in CAM. (ii) It is relatively easy to identify the spine index of new nodes as they are inserted into the ongoing structure, and hence the encoding problem is straightforward. (iii) A simple approximation to c-command can be stated in terms of dominance spines, offering the possibility of capturing many c-command relations in a CAM architecture.

To compute a dominance spine, the sentence processor only needs to track the current spine index. This index can be encoded on individual nodes. Once nodes are annotated with spine indices, an approximation to c-command can be stated as follows.

APPROXIMATION TO C-COMMAND

A pair of nodes whose parents share the same spine index stand in a c-command relation.

This heuristic accurately captures all c-command relations that it identifies. For example, in Figure 1 the heuristic successfully captures all of the nodes that are c-commanded by node C, i.e., D, E, F, G, and H, but not B or I and its daughters. Node C's parent, B, has a spine index of 2. The parents of nodes D, E, F, G, and H also have 2 as their index.

The dominance spine heuristic remains blind to some c-command relations: specifically, c-command relations in which the path from a c-commanding node to the c-commanded node traverses a left branch and fails to terminate at that point. For example, in Figure X, Node B c-commands nodes I, J, K, L, and M. The parents of nodes I, J, and K share a spine index with the parent of B, and hence the heuristic successfully identifies those c-command relations. But the parents of nodes L and M do not share a spine index with the parent of B. Thus, this heuristic does not capture the fact that B c-commands nodes L and M.

An advantage of dominance spines is that instead of initiating a c-command annotation process anew with every word and intermediate projection, as the previous approach required, the sentence processor only needs to keep in its control state a counter that represents the index of the current dominance spine. This bypasses the limited focus of attention considerations that the structure-as-content approach raised. As new nodes are created, the nodes are assigned a feature-value pair based on the current state of the counter. With every left-branching node, the spine index in the control state is incremented. Any new right-branching nodes that are created are assigned the new spine index.

As stated above, the heuristic depends on the spine index of the parents of the nodes that stand in a c-command relation, rather than on properties of the nodes themselves. Since it is not straightforward for retrieval cues in a CAM architecture to target memory chunks based on properties of other related chunks, implementation of this heuristic would probably require that each memory chunk explicitly encode its parent node's spine index. Since existing versions of the L&V model encode the identity of the parent node, there should be minimal cost in adding this new feature-attribute pair.

MODEL CHANGES AND EMPIRICAL CONSEQUENCES. This approach requires an additional buffer to keep track of the spine index. This does not significantly change the L&V model, as only a single item will be held in that buffer: the spine index.

The spine index proposal requires elaboration of the encoding of individual memory chunks, but as we have shown, this can be done by relying on information that is local to each individual node at the point when it is inserted into the structure. As such, it is consistent with the size-independent time cost of structural dependency formation. Similarly, since the heuristic implementation of c-command requires retrieval cues that simply entail matching between the parent-spine-index feature of the triggering node and the corresponding feature on the target node, it is consistent with the standard cue matching procedures of the L&V model.

The limitation of this heuristic is that it is predicted to be insensitive to some types of c-command relations. Specifically, the c-command relation includes relations in which the c-commandee is arbitrarily deeply embedded inside a left branch. For example, The quantified NPs in (9a-b), repeated from (4) above, differ in their ability to license a bound variable reading of the pronoun *he* in the subject position of the embedded clause. This reading is disallowed in (9b) because the NP *no physics professor* is embedded inside a relative clause, from where it

fails to c-command the pronoun. However, the embedding of the pronoun *his* in (9c-d) as the possessor of the embedded clause subject, or as the possessor of the possessor of the subject, has no negative effect on the availability of a bound variable reading. Native speakers of English treat (9 a, c, d) as all equally possible under a bound variable interpretation. But the heuristic implementation of c-command would incorrectly treat (9c-d) as cases where c-command does not obtain, grouping these together with (9b). We are unaware of empirical evidence that comprehenders are differentially sensitive to the availability of an antecedent in (9a) vs. (9c-d).

9. a. No student$_i$ [that listened to this physics professor] thinks that he$_i$ is a genius.

   b. * The student [that listened to no physics professor$_i$] thinks that he$_i$ is a genius.

   c. #No student$_i$ [that listened to this physics professor] thinks that his$_i$ TA is a genius.

   d. # No student$_i$ [that listened to this physics professor] thinks that his$_i$ classmate's TA is a genius.

5.4 Avoiding retrieval: Active maintenance of potential commanders

A fourth approach to the c-command problem is to assume that the parser is sensitive to c-command relations, but not via retrieval mechanisms. As we have discussed above in Section 4, there are some clear cases of on-line sensitivity to c-command constraints that might not directly implicate retrieval. Processing of filler-gap dependencies requires the parser to identify potential gap positions that are c-commanded by the filler phrase (e.g., Stowe 1986; Phillips & Wagers, 2007), and processing of backwards anaphora requires the parser to identify positions that are ineligible antecedent positions as they are c-commanded by the pronoun (e.g., Kazanina et al., 2007). However, cases such as these plausibly involve a forwards, prospective search for

the second member of a linguistic dependency. This could mean that the parser is able to carry forward the filler and its requirements, or the pronoun and its requirements, as part of the goal state of the parser. If this information is carried forward in a special buffer, and if the parser can successfully update its goals, depending on whether the current node is in the c-command path of the filler or pronoun, then it may be possible to construct these dependences in a c-command sensitive fashion without specifically requiring c-command to constrain retrieval.

Not all linguistic dependencies that respect c-command restrictions have the property of filler-gap dependencies that the left-edge of the dependency signals that a dependency is required. In cases of bound variable anaphora the need for the dependency is typically not apparent until a pronoun at the right-edge of the dependency is reached. Similarly, if NPI licensing is regarded as a case of an item-to-item dependency, the relevance of the licensor is generally not apparent until the NPI is reached at the right-edge of the dependency. Nevertheless, a possibility is that the parser might achieve c-command sensitivity via a generalization of the mechanism used for filler-gap dependencies. Potential licensors of c-command sensitive dependencies could be carried forward in a special buffer for as long as the parser is analyzing input that they c-command, thereby making them highly accessible when the dependent element is reached, but without invoking a c-command feature in retrieval. For example, a quantificational noun phrase, which might serve as the binder for subsequent pronouns, may be maintained in a privileged state. The same could be true for downward entailing operators that could serve as licensors for NPIs.

This generalization of active search mechanisms as a work-around to c-command constraints on retrieval is logically possible, but too little is known at present to assess its plausibility. First, the feasibility of the mechanism depends on the diversity of potential

c-commanders that would need to be stored in a special buffer in anticipation of use in possible dependencies. Quantificational NPs and downward-entailing operators such as negation are relatively easy to identify, but the range of potential participants in a c-command sensitive dependency is far broader. For example, all NPs, not just explicitly quantificational ones, can license bound-variable interpretations of pronouns, as shown by evidence from VP-ellipsis phenomena (Sag, 1976; Williams, 1977; Shapiro & Hestvik, 1995). A strategy that entailed maintaining all NPs in a c-command path in a special buffer is less attractive than one that does this only for quantificational NPs, which are rarer.

5.5 Denial: No on-line use of c-command in retrieval

A final possibility is that the parser is not, in fact, able to use c-command constraints in on-line retrieval processes. As we have discussed in Section 4 above, existing evidence on the role of c-command constraints in rapid on-line measures is limited, once we set aside cases that can be attributed to forwards-looking processes that might not implicate retrieval, e.g., constraints on filler-gap dependencies and backwards anaphora (Kazanina et al., 2007; Phillips, 2012), and if we also set aside cases where a local structural constraint such as *subject of the current clause* may serve as a proxy for a c-command restriction, e.g., constraints on reflexive licensing (Dillon et al., submitted; Nicol & Swinney, 1989; Patil et al., submitted; Sturt, 2003). If the parser cannot use c-command constraints, then the challenge for a CAM-based architecture might go away.

Although this option has some initial appeal, we should stress two reasons for treating it with skepticism. First, as described in Section 4, there are a number of findings that do suggest rapid on-line sensitivity to c-commanding items in memory. Recall that sensitivity to c-command

does not entail blindness to non c-commanding elements, so this is separate from questions about whether structural cues have a privileged status. Demonstrations that c-commanding elements have some advantage over non c-commanding elements are sufficient to show the on-line effects of a c-command constraint. Second, even if it were the case that rapid on-line retrieval processes were insensitive to relational notions such as c-command, this does not change the vast amount of evidence that speakers are highly sensitive to c-command constraints in their linguistic judgments. Therefore, if on-line retrieval were to use a memory architecture that cannot exploit c-command relations, then there would need to be a second architecture for language that is able to encode such relations.

5.6 Retreat: Some linguistic operations rely on a serial mechanism

A final option is to conclude that the options laid out above are either too cumbersome computationally or too limited empirically to capture human sensitivity to c-command relations. In this situation we might conclude that it is attractive to invoke a serial mechanism for accessing memory that explicitly uses the graph structure of a tree to access c-commanding nodes. Such mechanisms capture relational notions such as c-command with far greater ease than do mechanisms that implement parallel access. We should emphasize that this is not strictly a choice between a CAM-based architecture and a non-CAM architecture, as it is possible to carry out serial search in a CAM architecture. For example, a sequence of separate retrieval operations using individual node IDs as retrieval cues can be used to simulate a serial search of a tree. But this use of CAM sacrifices the constant time access property that has been argued to be a distinctive property of retrieval in human parsing. It also predicts greater immunity to interference. Although many empirical findings suggest the use of parallel access mechanism,

there are a number of empirical findings that suggest size-dependent, interference-insensitive on-line retrieval processes (see Dillon, 2011 for discussion).

6. Conclusion

The comprehension of any individual sentence is extended in time, and hence memory retrieval mechanisms play an important role in sentence processing. Numerous findings suggest that retrieval in sentence comprehension is carried out using a parallel access mechanism in the context of a content-addressable memory (CAM) architecture, in which retrieval cues are matched against all items in memory in parallel. This approach is well suited to retrieval operations that target inherent properties of memory chunks, such as animacy, subjecthood, person, or number. But it is less obviously well suited to retrieval operations that are subject to constraints involving configurational relations such as c-command, a relation that has a pervasive role in linguistic phenomena. We explored a number of ways in which c-command relations might be captured in a parallel access mechanism, and described their consequences for an explicit implementation of a CAM-based architecture proposed by Lewis and Vasishth (2005). None of the options provides a solution that perfectly fits the goals of the model or the existing empirical findings. C-command presents no such problems for serial mechanisms, which are also compatible with a CAM architecture. These considerations should be taken seriously in assessing the psychological plausibility of different schemes for encoding and navigating sentence structures.

Acknowledgments

This work was supported in part by NSF Grant BCS-0848554 to CP and NSF IGERT Grant DGE-0801465 to the University of Maryland. We thank Brian Dillon, John Hale, Dave Kush, Patrick Sturt, Titus van der Malsburg, Shravan Vasishth, and Matt Wagers for useful discussion of the material presented here. All errors are our own.

References

Alcocer, P., Maia, M., França, A., & Phillips, C. (2010). Structure sensitive and insensitive retrieval of subjects in Brazilian Portuguese. Talk at the 23rd annual CUNY Conference on Human Sentence Processing. New York, NY.

Aoshima, S., Yoshida, M., & Phillips, C. (2009). Incremental processing of coreference and binding in Japanese. *Syntax, 12*, 93-134.

Anderson, J. R. (2005). Human symbol manipulation within an integrated cognitive architecture. *Cognitive Science, 29*, 313-341.

Badecker, W. & Straub, K. (2002). The processing role of structural constraints on the interpretation of pronouns and anaphora. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 28*, 748-769.

Büring, D. (2005). *Binding theory*. Cambridge University Press.

Chomsky, N. (1981). *Lectures on government and binding*. Dordrecht: Foris.

Clackson, K., Felser, C., & Clahsen, H. (2011). Children's processing of reflexives and pronouns in English: Evidence from eye-movements during listening. *Journal of Memory and Language, 65, 128-144.*

Clifton, C. Frazier, L., & Deevy, P. (1999). Feature manipulation in sentence comprehension. *Rivista di Linguistica, 11*, 11-39.

Cowan, N. (2000). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences, 24*, 87-185.

Dillon, B. (2011). *Structured access in sentence comprehension.* PhD dissertation, University of Maryland.

Dillon et al. 2009 (?). SPR study on *ziji* and interference.

Dillon, B., Chow, W.-Y., Wagers, M., Guo, T., Liu, F., & Phillips, C. (submitted). The structure-sensitivity of memory access: Evidence from Mandarin Chinese.

Dillon, B., Mishler, A., Sloggett, S., & Phillips, C. (submitted). Contrasting intrusion profiles for agreement and anaphora: Experimental and modeling evidence.

Giannakidou, A. (2011). Negative and positive polarity items. In K. von Heusinger, C. Maienborn, & P. Portner (eds.), *Semantics: An international handbook of natural language meaning (2nd edition). Berlin: Mouton de Gruyter.*

Kazanina, N., Lau, E., Yoshida, M., Lieberman, M., & Phillips, C. (2007). The effect of syntactic constraints on the processing of backwards anaphora. *Journal of Memory and Language, 56*, 384-409.

Kazanina, N. & Phillips, C. (2010). Differential effects of constraints in the processing of Russian cataphora. *Quarterly Journal of Experimental Psychology, 63*, 371-400.

Kush, D., Lidz, J., & Phillips, C. (2012). Processing bound variable anaphora: Implications for memory encoding and retrieval. Talk at the Linguistic Society of America annual meeting, Portland, Oregon.

Lago, S., Alcocer, P., & Phillips, C. (2011). Agreement attraction in Spanish: Immediate vs. delayed sensitivity. Poster presentation at the 24th annual CUNY Conference on Human Sentence Processing. Stanford, CA.

Lewis, R. L. & Vasishth, S. (2005). An activation-based model of sentence comprehension as skilled memory retrieval. *Cognitive Science, 29*, 375-419.

Lewis, R. L., Vasishth,, S., & Van Dyke, J. A. (2006). Computational principles of working memory in sentence comprehension. *Trends in Cognitive Sciences, 10*, 447-454.

McElree, B. (2006). Accessing recent events. In B. H. Ross (ed.), *The psychology of learning and motivation* (Vol 46), pp. 155-200. San Diego: Academic Press.

McElree, B., Foraker, S., & Dyer, L. (2003). Memory structures that subserve sentence comprehension. *Journal of Memory and Language, 48*, 67-91.

Martin, A. E. & McElree, B. (2008). A content-addressable pointer mechanism underlies comprehension of verb phrase ellipsis. *Journal of Memory and Language, 58*, 879-906.

Martin, A. E. & McElree, B. (2009). Memory operations that support language comprehension: evidence from verb-phrase ellipsis. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 35*, 1231-1239.

Nicol, J. & Swinney, D. (1989). The role of structure in coreference assignment during sentence comprehension. *Journal of Psycholinguistic Research, 18*, 5-19.

Parker, D., Lago, S., & Phillips, C. (2012). Retrieval interference in the resolution of anaphoric PRO. Talk at the 35th annual GLOW conference, Potsdam, Germany.

Parker, D. & Phillips, C. (2011). Illusory negative polarity item licensing is selective. Poster presentation at the 24th annual CUNY Conference on Human Sentence Processing. Stanford, CA.

Patil, U., Vasishth, S., & Lewis, R. L. (submitted). Retrieval interference in syntactic processing: The case of reflexive binding in English.

Pearlmutter, N. J., Garnsey, S., M., & Bock, K. (1999). Agreement processes in sentence comprehension. *Journal of Memory and Language, 41*, 427-456.

Phillips, C. (2012). On the nature of island constraints. I: Language processing and reductionist accounts. In J. Sprouse & N. Hornstein (eds.), *Experimental syntax and island effects.* Cambridge University Press.

Phillips, C., Wagers, M. W., & Lau, E. F. (2011). Grammatical illusions and selective fallibility in real-time language comprehension. In J. Runner (ed.), *Experiments at the Interfaces*, *Syntax & Semantics, vol. 37*, pp. 147-180. Bingley, UK: Emerald.

Reinhart, T. (1983). *Anaphora and semantic interpretation*. London: Croom Helm.

Sag, I. A. (1976). *Deletion and logical form*. PhD dissertation, MIT.

Shapiro, L. P. & Hestvik, A. (1995). On-line comprehension of VP-ellipsis: Syntactic reconstruction and semantic influence. *Journal of Psycholinguistic Research, 24*, 517-532.

Staub, A. (2009). On the interpretation of the number attraction effect: Response time evidence. *Journal of Memory and Language, 60*, 308-327.

Sturt, P. (2003). The time course of application of binding constraints in reference resolution. *Journal of Memory and Language, 48*, 542-562.

Van Dyke, J. A. & McElree, B. (2006). Retrieval interference in sentence comprehension. *Journal of Memory and Language, 55*, 157-166.

Vasishth, S., Brüssow, S., Lewis, R., & Drenhaus, H. (2008). Processing polarity: How the ungrammatical intrudes on the grammatical. *Cognitive Science, 32*, 685-712.

Wagers, M. W., Lau, E. F., & Phillips, C. (2009). Agreement attraction in comprehension: Representations and processes. *Journal of Memory and Language, 61*, 206-237.

Williams, E. (1977). Discourse and logical form. *Linguistic Inquiry, 8*, 101-139.

Xiang, M., Dillon, B., & Phillips, C. (2006). Testing the strength of the spurious licensing effect for negative polarity items. Talk at the 19th annual CUNY Conference on Human Sentence Processing. New York City.

Xiang, M., Dillon, B., & Phillips, C. (2009). Illusory licensing effects across dependency types: ERP evidence. *Brain and Language, 108*, 40-55.