Cognitive Compositional Semantics using Continuation Dependencies

William Schuler
Department of Linguistics
The Ohio State University
schuler@ling.osu.edu

Adam Wheeler
Department of Linguistics
The Ohio State University
wheeler@ling.osu.edu

Abstract

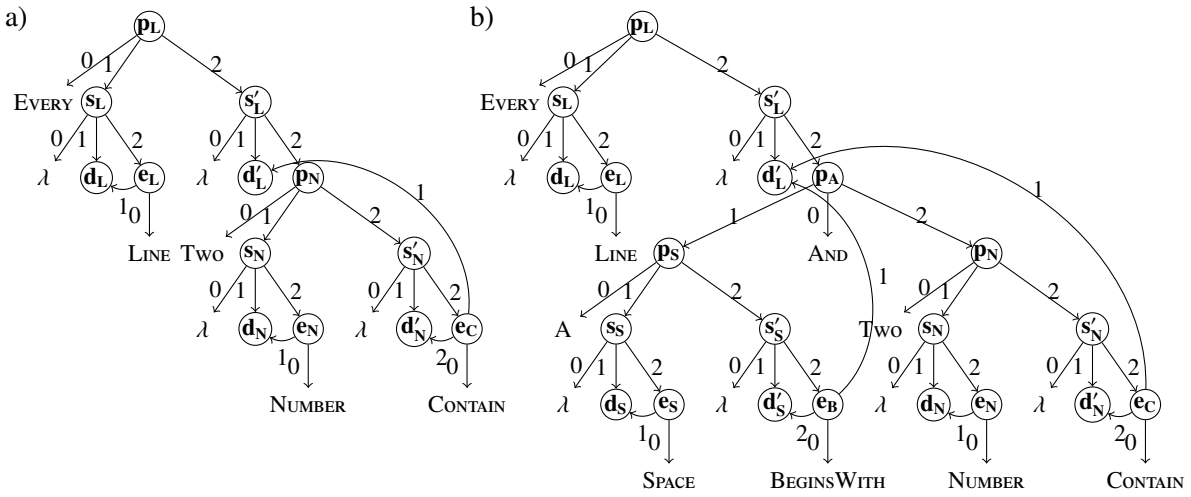
This paper describes a graphical semantic representation based on bottom-up 'continuation' dependencies which has the important property that its vertices define a usable set of discourse referents in working memory even in contexts involving conjunction in the scope of quantifiers. An evaluation on an existing quantifier scope disambiguation task shows that non-local continuation dependencies can be as reliably learned from annotated data as representations used in a state-of-the-art quantifier scope resolver, suggesting that continuation dependencies may provide a natural representation for scope information.

1 Introduction

It is now fairly well established that at least shallow semantic interpretation informs parsing decisions in human sentence processing (Tanenhaus et al., 1995; Brown-Schmidt et al., 2002), and recent evidence points to incremental processing of quantifier implicatures as well (Degen and Tanenhaus, 2011). This may indicate that inferences about the meaning of quantifiers are processed directly in working memory. Human working memory is widely assumed to store events (including linguistic events) as re-usable activation-based states, connected by a durable but rapidly mutable weight-based memory of cued associations (Marr, 1971; Anderson et al., 1977; Murdock, 1982; McClelland et al., 1995; Howard and Kahana, 2002). Complex dependency structures can therefore be stored in this associative memory as graphs, with states as vertices and cued associations as directed edges (e.g. Kintsch, 1988). This kind of representation is necessary to formulate and evaluate algorithmic claims (Marr, 1982) about cued associations and working memory use in human sentence processing (e.g. van Schijndel and Schuler, 2013). But accounting for syntax and semantics in this way must be done carefully in order to preserve linguistically important distinctions. For example, positing spurious local dependencies in filler-gap constructions can lead to missed integrations of dependency structure in incremental processing, resulting in weaker model fitting (van Schijndel et al., 2013). Similar care may be necessary in cases of dependencies arising from anaphoric coreference or quantifier scope.

Unfortunately, most existing theories of compositional semantics (Montague, 1973; Barwise and Cooper, 1981; Bos, 1996; Baldridge and Kruijff, 2002; Koller, 2004; Copestake et al., 2005) are defined at the *computational* level (Marr, 1982), employing beta reduction over complete or underspecified lambda calculus expressions as a precise description of the language processing task to be modeled, not at the *algorithmic* level, as a model of human language processing itself. The structured expressions these theories generate are not intended to represent re-usable referential states of the sort that could be modeled in current theories of associative memory. As such, it should not be surprising that structural adaptations of lambda calculus expressions as referential states exhibit a number of apparent deficiencies:

First, representations based on lambda calculus expressions lack topologically distinguishable referents for sets defined in the context of outscoping quantifiers. For example, a structural adaptation of a lambda calculus expression for the sentence *Every line contains two numbers,* shown in Figure 1a (adapted from Koller, 2004), contains referents for the set of all document lines ($\mathbf{s_L}$) and for the set of all numbers ($\mathbf{s_N}$) which can be identified by cued associations to predicate constants like NUMBER, but it is not clear how a referent for the set of numbers in document lines can be distinguished from a referent for the set of numbers



c) $(\textsc{Every}\ \mathbf{p_L}\ \mathbf{s_L}\ \mathbf{s'_L}) \wedge (\textsc{Set}\ \mathbf{s_L}\ \mathbf{d_L}\ \mathbf{e_L}) \wedge (\textsc{Line}\ \mathbf{e_L}\ \mathbf{d_L}) \wedge (\textsc{Set}\ \mathbf{s'_L}\ \mathbf{d'_L}\ \mathbf{p_N}) \wedge$
$(\textsc{Two}\ \mathbf{p_N}\ \mathbf{s_N}\ \mathbf{s'_N}) \wedge (\textsc{Set}\ \mathbf{s_N}\ \mathbf{d_N}\ \mathbf{e_N}) \wedge (\textsc{Number}\ \mathbf{e_N}\ \mathbf{d_N}) \wedge (\textsc{Set}\ \mathbf{s'_N}\ \mathbf{d'_N}\ \mathbf{e_C}) \wedge (\textsc{Contain}\ \mathbf{e_C}\ \mathbf{d'_L}\ \mathbf{d'_N})$

Figure 1: Semantic dependency graph in a 'direct' (top-down) style, adapted from a disambiguated representation of Koller (2004), excluding quantifiers over eventualities. The semantic dependency structure for the sentence *Every line contains two numbers* (a), with flat logical form (c), is not a subgraph of the semantic dependency structure for *Every line begins with a space and contains two numbers* (b), because the structure is interrupted by the explicit conjunction predicate 'AND'.

in *each* document line ($\mathbf{s'_N}$) using local topological features of the dependency graph, as would be required to accurately recall assertions about total or average quantities of numbers in document lines.[1]

Second, graphs based on traditional lambda calculus representations do not model conjuncts as subgraphs of conjunctions. For example, the graphical representation of the sentence *Every line begins with a space and contains two numbers* shown in Figure 1b does not contain the graphical representation of the sentence *Every line contains two numbers* shown in Figure 1a as a connected subgraph. Although one might expect a query about a conjunct to be directly answerable from a knowledge base containing the conjoined representation, the pattern of dependencies that make up the conjunct in a graphical representation of a lambda calculus expression does not match those in the larger conjunction.

Finally, representations based on lambda calculus expressions contain vertices that do not seem to correspond to viable discourse referents. For example, following the sentence *Every line contains two numbers,* using the lambda expression shown in Figure 1b, $\mathbf{d_L}$ may serve as a referent of *it* in *but it has only one underscore,* $\mathbf{s_N}$ may serve as a referent of *they* in *but they are not negative,* $\mathbf{e_C}$ may serve as a referent of *that* in *but that was before it was edited,* and $\mathbf{p_L}$ may serve as a referent of *that* in *but the compiler doesn't enforce that,* but it is not clear what if anything would naturally refer to the internal conjunction $\mathbf{p_A}$. Predications over such conjunctions (e.g. *Kim believes that every line begins with a space and contains*

[1] This graph matching can be implemented in a vectorial model of associative memory by comparing the (e.g. cosine) similarity of superposed vectors resulting from cueing incoming and outgoing dependencies with all possible labels in increasingly longer paths from one or more constant vector states (e.g. vectors for predicate constants). This graph matching does not necessarily preclude the introduction of monotonicity constraints from matched quantifiers. For example, *More than two perl scripts work,* can entail *More than two scripts work,* using a subgraph in the first argument, but *Fewer than two scripts work,* can entail *Fewer than two perl scripts work,* using a supergraph in the first argument. This consideration is similar to those observed in representations based on natural logic (MacCartney and Manning, 2009) which also uses low-level matching to perform some kinds of inference, but representations based on natural logic typically exclude other forms of inference, whereas the present model does not.

This matching also assumes properties of nuclear scope variables are inherited from associated restrictor variables, e.g. through a set of dependencies from nuclear scope sets to restrictor sets not shown in the figure. This assumption will be revisited in Section 3.

two numbers) are usually predicated at the outer proposition $\mathbf{p_L}$, and in any case do not have truth values that are independent of the same predication at each conjunct. One of the goals of Minimal Recursion Semantics (Copestake et al., 2005) was to eliminate similar kinds of superfluous conjunction structure.

Fortunately, lambda calculus expressions like those shown in Figure 1 are not the only way to represent compositional semantics of sentences. This paper defines a graphical semantic dependency representation that can be translated into lambda calculus, but has the important property that its vertices define a usable set of discourse referents in working memory even in contexts involving conjunction in the scope of quantifiers. It does this by reversing the direction of dependencies from parent-to-child subsumption in a lambda-calculus tree to a representation similar to the inside-out structure of function definitions in a continuation-passing style (Barker, 2002; Shan and Barker, 2006)[2] so that sets are defined in terms of their context, and explicit 'AND' predicates are no longer required, leaving nothing to get in the way of an exact pattern match.[3] The learnability of the non-local continuation dependencies involved in this representation is then evaluated on an existing quantifier scope disambiguation task using a dependency-based statistical scope resolver, with results comparable to a state-of-the-art unrestricted graph-based quantifier scope resolver (Manshadi et al., 2013).

2 Continuation Dependencies

This paper explores the use of a bottom-up dependency representation, inspired by the inside-out structure of function definitions in a continuation-passing style (Barker, 2002; Shan and Barker, 2006), which creates discourse referents for sets that are associated with particular scoping contexts. This dependency representation preserves the propositions, sets, eventualities, and ordinary discourse referents of a 'direct' representation (the **p**, **s**, **e**, and **d** nodes in Figure 1), but replaces the downward dependencies departing set referents with upward dependencies to context sets (highlighted in Figure 2).

Figures 1c and 2c also show flat logical forms composed of *elementary predications*, adapted from Kruijff (2001) and Copestake et al. (2005), for the sentence *Every line contains two numbers,* which are formed by identifying the function associated with the predicate constant (e.g. CONTAIN) that is connected to each proposition or eventuality referent (e.g. $\mathbf{e_C}$) by a dependency labeled '0', then applying that function to this referent, followed by the list of arguments connected to this referent by functions numbered '1' and up: e.g. (CONTAIN $\mathbf{e_C}$ $\mathbf{d'_L}$ $\mathbf{d'_N}$). These dependencies can also be defined by numbered dependency functions $\mathbf{f}_n$ from source instance j to destination instance i, notated $(\mathbf{f}_n\ j) = i$. This notation will be used in Section 4 to define constraints in the form of equations. For example, the subject (first argument) of a lexical item may be constrained to be the subject (first argument) of that item's sentential complement (second argument), as in an instance of subject control, using the dependency equation $(\mathbf{f}_1\ i) = (\mathbf{f}_1\ (\mathbf{f}_2\ i))$.

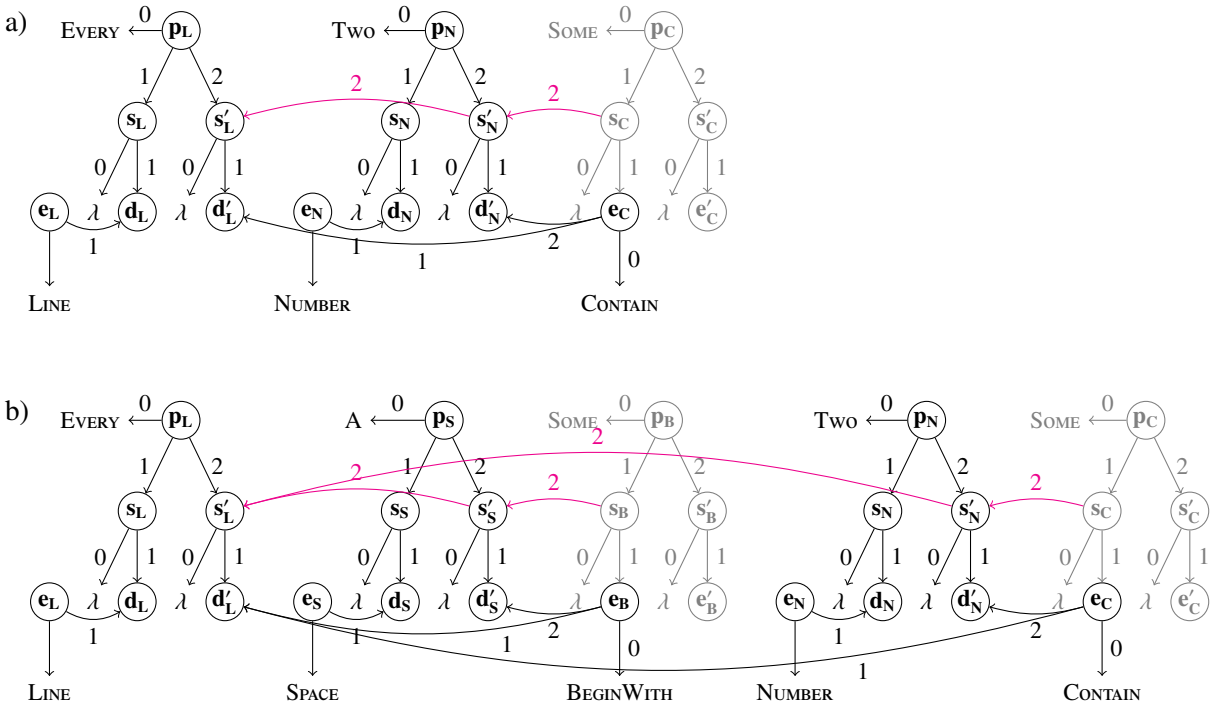
Since continuation dependencies all flow up the tree, any number of conjuncts can impinge upon a common outscoping continuation, so there is no longer any need for explicit conjunction nodes. The representation is also attractive in that it locally distinguishes queries about, say, the cardinality of the set of numbers in each document line (SET $\mathbf{s'_N}$ $\mathbf{d'_N}$ $\mathbf{s'_L}$) from queries about the cardinality of the set of numbers in general (SET $\mathbf{s'_N}$ $\mathbf{d'_N}$ $\mathbf{s'_\perp}$) which is crucial for successful inference by pattern matching. Finally, connected sets of continuation dependencies form natural 'scope graphs' for use in graph-based disambiguation algorithms (Manshadi and Allen, 2011; Manshadi et al., 2013), which will be used to evaluate this representation in Section 6.

3 Mapping to Lambda Calculus

It is important for this representation not only to have attractive graphical subsumption properties, but also to be sufficiently expressive to define corresponding expressions in lambda calculus. When continuation dependencies are filled in, the resulting dependency structure can be trans-

[2] This representation also has much in common with generalized Skolem terms of Steedman (2012), which also represent dependencies to outscoped terms, but here continuation dependencies are applied to all quantifiers, including universals.

[3] This also holds for explicit disjunction predicates, which can be cast as conjunction through application of de Morgan's law and manipulation of the polarity of adjacent quantifiers. For example, *Every line begins with at least one space or contains at least two numbers,* is equivalent to *No line begins with fewer than one space and contains fewer than two numbers.*



c) $(\textsc{Every}\ \mathbf{p_L}\ \mathbf{s_L}\ \mathbf{s'_L}) \wedge (\textsc{Set}\ \mathbf{s_L}\ \mathbf{d_L}\ \mathbf{s}_\perp) \wedge (\textsc{Line}\ \mathbf{e_L}\ \mathbf{d_L}) \wedge (\textsc{Set}\ \mathbf{s'_L}\ \mathbf{d'_L}\ \mathbf{s}_\perp) \wedge$
$(\textsc{Two}\ \mathbf{p_N}\ \mathbf{s_N}\ \mathbf{s'_N}) \wedge (\textsc{Set}\ \mathbf{s_N}\ \mathbf{d_N}\ \mathbf{s}_\perp) \wedge (\textsc{Number}\ \mathbf{e_N}\ \mathbf{d_N}) \wedge (\textsc{Set}\ \mathbf{s'_N}\ \mathbf{d'_N}\ \mathbf{s'_L}) \wedge (\textsc{Contain}\ \mathbf{e_C}\ \mathbf{d'_L}\ \mathbf{d'_N})$

Figure 2: Semantic dependency graph in a 'continuation-passing' (bottom-up) style, including quantifiers over eventualities for verbs (in gray). The semantic dependency structure for the sentence *Every line contains two numbers* (a), with flat logical form (c), is now contained by the semantic dependency structure for *Every line begins with a space and contains two numbers* (b).

lated into a lambda calculus expression by a deterministic algorithm which traverses sequences of continuation dependencies and constructs accordingly nested terms in a manner similar to that defined for DRT (Kamp, 1981). This graphical representation can be translated into lambda calculus by representing the source graph as a set Γ of elementary predications $(f\ i_0\ ..\ i_N)$ and the target as a set Δ of translated lambda calculus expressions, e.g. $(\lambda_i\ (h_f\ i_0\ ..\ i\ ..\ i_N))$. The set Δ can then be derived from Γ using the following natural deduction rules:[4]

- Initialize Δ with lambda terms (sets) that have no outscoped sets in Γ:

$$\frac{\Gamma, (\textsc{Set}\ s\ i\ \_\ \_)\ ;\ \Delta}{\Gamma, (\textsc{Set}\ s\ i\ \_\ \_)\ ;\ (\lambda_i\ \textsc{True}), \Delta}\ (\textsc{Set}\ \_\ \_\ s\ \_) \notin \Gamma$$

- Add constraints to appropriate sets in Δ:

$$\frac{\Gamma, (f\ i_0\ ..\ i\ ..\ i_N)\ ;\ (\lambda_i\ o), \Delta}{\Gamma\ ;\ (\lambda_i\ o \wedge (h_f\ i_0\ ..\ i\ ..\ i_N)), \Delta}\ i_0 \in E$$

- Add constraints of supersets as constraints on subsets in Δ:

$$\frac{\begin{array}{l}\Gamma, (\textsc{Set}\ s\ i\ \_\ \_), (\textsc{Set}\ s'\ i'\ s''s)\ ; \\ (\lambda_i\ o \wedge (h_f\ i_0\ ..\ i\ ..\ i_N)), (\lambda_{i'}\ o'), \Delta\end{array}}{\begin{array}{l}\Gamma, (\textsc{Set}\ s\ i\ \_\ \_), (\textsc{Set}\ s'\ i'\ s''s)\ ; \\ (\lambda_i\ o \wedge (h_f\ i_0\ ..\ i\ ..\ i_N)), \\ \qquad (\lambda_{i'}\ o' \wedge (h_f\ i_0\ ..\ i'\ ..\ i_N)), \Delta\end{array}}$$

- Add quantifiers over completely constrained sets in Δ:

$$\frac{\begin{array}{l}\Gamma, (\textsc{Set}\ s\ i\ \_\ \_), (f\ p\ s'\ s''), \\ \quad (\textsc{Set}\ s'\ i'\ s\ \_), (\textsc{Set}\ s''i''s'\ s')\ ; \\ (\lambda_i\ o), (\lambda_{i'}\ o'), (\lambda_{i''}\ o''), \Delta\end{array}}{\begin{array}{l}\Gamma, (\textsc{Set}\ s\ i\ \_\ \_)\ ; \\ (\lambda_i\ o \wedge (h_f\ (\lambda_{i'}\ o')\ (\lambda_{i''}\ o''))), \Delta\end{array}}\ \begin{array}{l}p \in P, \\ (f'..\ i'..) \notin \Gamma, \\ (f''..\ i''..) \notin \Gamma.\end{array}$$

For example, the graph in Figure 2 can be translated into the following lambda calculus expression (including quantifiers over eventualities in the source graph, to eliminate unbound variables):

[4]Here, set predications are defined with an additional final argument position, which is defined to refer in a nuclear scope set to the restrictor set that is its sibling, and in a restrictor set to refer to $\mathbf{s}_\perp$.

$$(\textsc{Every}\ (\lambda_{d_L}\textsc{Some}\ (\lambda_{e_L}\textsc{Line}\ e_L\ d_L))\ (\lambda_{d'_L}\textsc{Two}\ (\lambda_{d_N}\textsc{Some}\ (\lambda_{e_N}\textsc{Number}\ e_N\ d_N))\ (\lambda_{d'_N}\textsc{Some}\ (\lambda_{e_C}\textsc{Contain}\ e_C\ d'_L\ d'_N))))$$

4 Derivation of Syntactic and Semantic Dependencies

The semantic dependency representation defined in this paper assumes semantic dependencies other than those representing continuations are derived compositionally by a categorial grammar. In particular, this definition assumes a Generalized Categorial Grammar (GCG) (Bach, 1981; Oehrle, 1994), because it can be used to distinguish argument and modifier compositions (from which restrictor and nuclear scope sets are derived in a tree-structured continuation graph), and because large GCG-annotated corpora defined with this distinction are readily available (Nguyen et al., 2012). GCG category types $c \in C$ each consist of a primitive category type $u \in U$, typically labeled with the part of speech of the head of a category (e.g. **V**, **N**, **A**, etc., for phrases or clauses headed by verbs, nouns, adjectives, etc.), followed by one or more unsatisfied dependencies, each consisting of an operator $o \in O$ (**-a** and **-b** for adjacent argument dependencies preceding and succeeding a head, **-c** and **-d** for adjacent conjunct dependencies preceding and succeeding a head, **-g** for filler-gap dependencies, **-r** for relative pronoun dependencies, and some others), each followed by a dependent category type from C. For example, the category type for a transitive verb would be **V-aN-bN**, since it is headed by a verb, and has unsatisfied dependencies to satisfied noun-headed categories preceding and succeeding it (for the subject and direct object noun phrase, respectively). This formulation has the advantage for semantic dependency calculation that it distinguishes modifier and argument attachment. Since the semantic representation described in this paper makes explicit distinctions between restrictor sets and scope sets (which is necessary for coherent interpretation of quantifiers) it is necessary to consistently apply predicate-argument constraints to discourse referents in the nuclear scope set of a quantifier and modifier-modificand constraints to discourse referents in the restrictor set of a quantifier. For example, in Sentence 1:

(1) Everything is [**A-aN** open].

the predicate *open* constrains the nuclear scope set of *every*, but in Sentence 2:

(2) Everything [**A-aN** open] is finished.

the predicate *open* constrains the restrictor set. These constraints can be consistently applied in the argument and modifier attachment rules of a GCG.

Like a Combinatory Categorial Grammar (Steedman, 2000), a GCG defines syntactic dependencies for compositions that are determined by the number and kind of unsatisfied dependencies of the composed category types. These are similar to dependencies for subject, direct object, preposition complement, etc., of Stanford dependencies (de Marneffe et al., 2006), but are reduced to numbers based on the order of the associated dependencies in the category type of the lexical head.

These syntactic dependencies are then associated with semantic dependencies, with the referent of a subject associated with the first argument of an eventuality, the referent of a direct object associated with the second argument, and so on, for all verb forms other than passive verbs. In the case of passive verbs, the referent of a subject is associated with the second argument of an eventuality, the referent of a direct object associated with the third argument, and so on.

In order to have a consistent treatment of argument and modifier attachment across all category types, and also in order to model referents of verbs as eventualities which can be quantified by adverbs like *never, once, twice,* etc. (Parsons, 1990), it is desirable for eventualities associated with verbs to also be quantified. Outgoing semantic dependencies to arguments of eventualities are then applied as constraints to the discourse referent variable of the restrictor sets of these quantifiers. Incoming dependencies to eventualities and other discourse referents used as modificands of modifiers are also applied as constraints to discourse referent variables of restrictor sets, but incoming dependencies to discourse referents used as arguments of predicates are applied as constraints to discourse referent variables of nuclear scope sets. This assignment to restrictor or nuclear scope sets depends on the context of the relevant (argument or modifier attachment) parser operation, so associations between syntactic and semantic dependencies must be left partially undefined in lexical entries. Lexical entries are therefore defined with separate syntactic and semantic dependencies, using even numbers for syntactic dependencies from lexical items, and odd numbers for

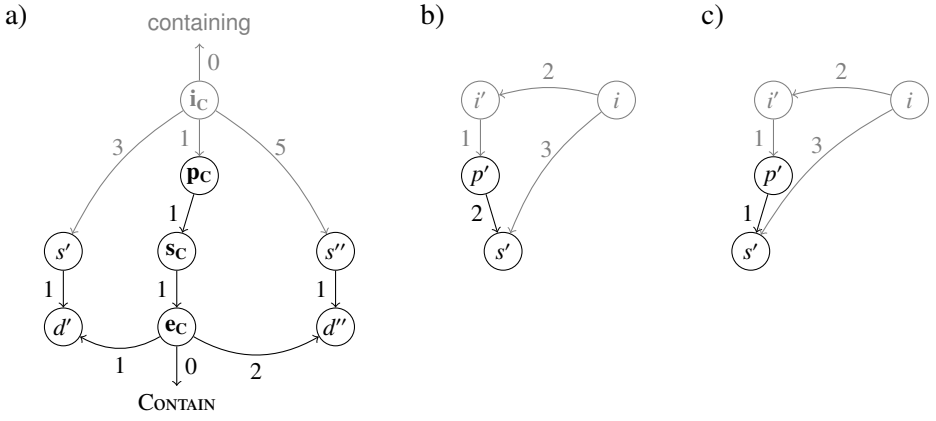


Figure 3: Example lexical semantic dependencies for the verb *containing* (a), and dependency equations for argument attachment (b) and modifier attachment (c) in GCG deduction rules. Lexical dependencies are shown in gray. Even numbered edges departing lexical items denote lexical syntactic dependencies, and odd numbered edges departing lexical items are lexical semantic dependencies. Argument attachments constrain semantic arguments to the nuclear scope sets of syntactic arguments, and modifier attachments constrain semantic arguments to the restrictor sets of syntactic arguments.

semantic dependencies from lexical items. For example, a lexical mapping for the finite transitive verb *contains* might be associated with the predicate CONTAIN, and have the discourse referent of its first lexical semantic argument ($\mathbf{f_1}$ ($\mathbf{f_3}$ i)) associated with the first argument of the eventuality discourse referent of the restrictor set of its proposition ($\mathbf{f_1}$ ($\mathbf{f_1}$ ($\mathbf{f_1}$ ($\mathbf{f_1}$ i)))), and the discourse referent of its second lexical semantic argument ($\mathbf{f_1}$ ($\mathbf{f_5}$ i)) associated with the second argument of the eventuality discourse referent of the restrictor set of its proposition ($\mathbf{f_2}$ ($\mathbf{f_1}$ ($\mathbf{f_1}$ ($\mathbf{f_1}$ i)))):

$$\begin{aligned}\text{contains} \Rightarrow\ & \mathbf{V\text{-}aN\text{-}bN} : \lambda_i\,(\mathbf{f_0}\ i)=\text{contains} \\ & \wedge\,(\mathbf{f_0}\ (\mathbf{f_1}\ (\mathbf{f_1}\ (\mathbf{f_1}\ i))))=\textsc{Contain} \\ & \wedge\,(\mathbf{f_1}\ (\mathbf{f_1}\ (\mathbf{f_1}\ (\mathbf{f_1}\ i))))=(\mathbf{f_1}\ (\mathbf{f_3}\ i)) \\ & \wedge\,(\mathbf{f_2}\ (\mathbf{f_1}\ (\mathbf{f_1}\ (\mathbf{f_1}\ i))))=(\mathbf{f_1}\ (\mathbf{f_5}\ i))\end{aligned}$$

A graphical representation of these dependencies is shown in Figure 3a. These lexical semantic constraints are then associated with syntactic dependencies by grammar rules for argument and modifier attachment, as described below.

4.1 Inference rules for argument attachment

In GCG, as in other categorial grammars, inference rules for argument attachment apply functors of category c-**a**d or c-**b**d to preceding or succeeding arguments of category d:

$$d : g \quad c\text{-}\mathbf{a}d : h \Rightarrow\ c : (\mathbf{f}_{c\text{-}\mathbf{a}d}\ g\ h) \qquad \text{(Aa)}$$

$$c\text{-}\mathbf{b}d : g \quad d : h \Rightarrow\ c : (\mathbf{f}_{c\text{-}\mathbf{b}d}\ g\ h) \qquad \text{(Ab)}$$

where $\mathbf{f}_{u\varphi_1\ldots\varphi_n}$ are composition functions for $u \in U$ and $\varphi \in \{\mathbf{-a}, \mathbf{-b}, \mathbf{-c}, \mathbf{-d}\} \times C$, which connect the lexical item ($\mathbf{f}_{2n}$ i) of a preceding child function g as the $2n^{\text{th}}$ argument of lexical item i of a succeeding child function h, or vice versa:

$$\begin{aligned}\mathbf{f}_{u\varphi_{1..n-1}\text{-}\mathbf{a}d} \stackrel{\text{def}}{=}\ & \lambda_{g\,h\,i}\,(g\,(\mathbf{f}_{2n}\ i)) \wedge (h\ i) \\ & \wedge\,(\mathbf{f}_{2n+1}\ i)=(\mathbf{f_2}\ (\mathbf{f_1}\ (\mathbf{f}_{2n}\ i))) \qquad \text{(1a)}\end{aligned}$$

$$\begin{aligned}\mathbf{f}_{u\varphi_{1..n-1}\text{-}\mathbf{b}d} \stackrel{\text{def}}{=}\ & \lambda_{g\,h\,i}\,(g\ i) \wedge (h\ (\mathbf{f}_{2n}\ i)) \\ & \wedge\,(\mathbf{f}_{2n+1}\ i)=(\mathbf{f_2}\ (\mathbf{f_1}\ (\mathbf{f}_{2n}\ i))) \qquad \text{(1b)}\end{aligned}$$

as shown in Figure 3b. This associates the lexical semantic argument of the predicate ($\mathbf{f}_{2n+1}$ i) with the nuclear scope of the quantifier proposition associated with the syntactic argument ($\mathbf{f_2}$ ($\mathbf{f_1}$ ($\mathbf{f}_{2n}$ i))). For example, the following inference attaches a subject to a verb:

$$\frac{\dfrac{\text{every line}}{\mathbf{N} : \lambda_i\,(\mathbf{f_0}\ i)=\text{line ..}} \quad \dfrac{\text{contains two numbers}}{\mathbf{V\text{-}aN} : \lambda_i\,(\mathbf{f_0}\ i)=\text{contains ..}}}{\begin{array}{c}\mathbf{V} : \lambda_i\,(\mathbf{f_0}\ (\mathbf{f_2}\ i))=\text{line .. } \wedge\,(\mathbf{f_0}\ i)=\text{contains ..} \\ \wedge\,(\mathbf{f_3}\ i)=(\mathbf{f_2}\ (\mathbf{f_1}\ (\mathbf{f_2}\ i)))\end{array}}\ \text{Aa}$$

4.2 Inference rules for modifier attachment

This grammar also uses distinguished inference rules for modifier attachment. Inference rules for modifier attachment apply preceding or succeeding modifiers of category u-**a**d to modificands of category c, for $u \in U$ and $c, d \in C$:

$$u\text{-}\mathbf{a}d : g \quad c : h \Rightarrow\ c : (\mathbf{f}_{\text{PM}}\ g\ h) \qquad \text{(Ma)}$$

$$c : g \quad u\text{-}\mathbf{a}d : h \Rightarrow\ c : (\mathbf{f}_{\text{SM}}\ g\ h) \qquad \text{(Mb)}$$

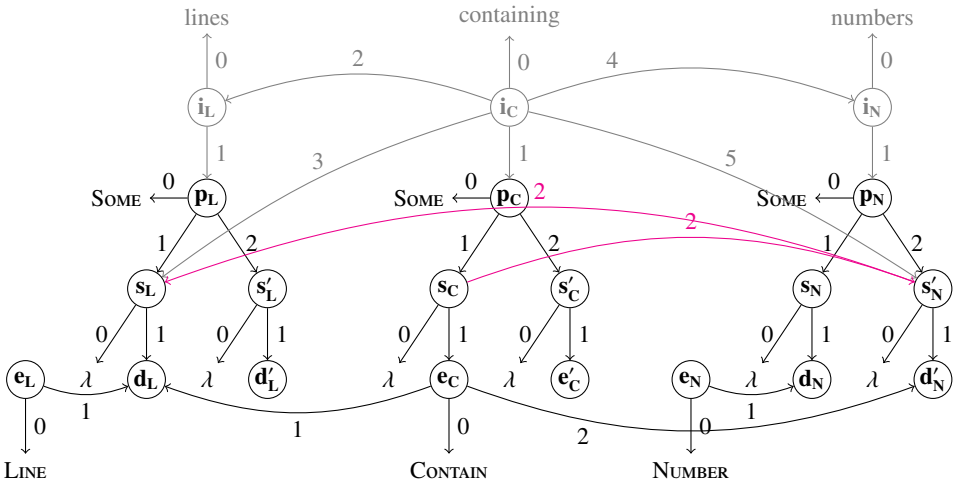


Figure 4: Compositional analysis of noun phrase *lines containing numbers* exemplifying both argument attachment (to *numbers*) and modifier attachment (to *lines*). Lexical dependencies are shown in gray, and continuation dependencies (which do not result from syntactic composition) are highlighted.

where $\mathbf{f}_{\text{PM}}$ and $\mathbf{f}_{\text{SM}}$ are category-independent composition functions for preceding and succeeding modifiers, which return the lexical item of the argument (j) rather than of the predicate (i):

$$\mathbf{f}_{\text{PM}} \stackrel{\text{def}}{=} \lambda_{g\,h\,j}\, \exists_i\, (\mathbf{f_2}\ i){=}j \wedge (g\ i) \wedge (h\ j) \wedge (\mathbf{f_3}\ i){=}(\mathbf{f_1}\ (\mathbf{f_1}\ (\mathbf{f_2}\ i))) \quad (2a)$$

$$\mathbf{f}_{\text{SM}} \stackrel{\text{def}}{=} \lambda_{g\,h\,j}\, \exists_i\, (\mathbf{f_2}\ i){=}j \wedge (g\ j) \wedge (h\ i) \wedge (\mathbf{f_3}\ i){=}(\mathbf{f_1}\ (\mathbf{f_1}\ (\mathbf{f_2}\ i))) \quad (2b)$$

as shown in Figure 3c. This allows categories for predicates to be re-used as modifiers. Unlike argument attachment, modifier attachment associates the lexical semantic argument of the modifier ($\mathbf{f}_{2n+1}\ i$) with the restrictor of the quantifier proposition of the modificand ($\mathbf{f_1}\ (\mathbf{f_1}\ (\mathbf{f}_{2n}\ i))$). For example, the following inference attaches an adjectival modifier to the quantifier proposition of a noun phrase:

$$\frac{\dfrac{\text{every line}}{\mathbf{N}{:}\lambda_i\ (\mathbf{f_0}\ i){=}\text{line}\ ..} \quad \dfrac{\text{containing two numbers}}{\mathbf{A\text{-}aN}{:}\lambda_i\ (\mathbf{f_0}\ i){=}\text{containing}\ ..}}{\begin{array}{c}\mathbf{N}:\lambda_i\ (\mathbf{f_0}\ i){=}\text{line}\ ..\ \wedge\ \exists_j\ (\mathbf{f_0}\ j){=}\text{containing}\ .. \\ \wedge\ (\mathbf{f_2}\ j){=}i \wedge (\mathbf{f_3}\ j){=}(\mathbf{f_1}\ (\mathbf{f_1}\ (\mathbf{f_2}\ j)))\end{array}}\ \text{Mb}$$

An example of argument and modifier attachment is shown in Figure 4.

5 Estimation of Scope Dependencies

Semantic dependency graphs obtained from GCG derivations as described in Section 4 are scopally underspecified. Scope disambiguations must then be obtained by specifying continuation dependencies from every set referent to some other set referent (or to a null context, indicating a top-level set). In a sentence processing model, these nonlocal continuation dependencies would be incrementally calculated in working memory in a manner similar to coreference resolution.[5] However, in this paper, in order to obtain a reasonable estimate of the learnability of such a system, continuation dependencies are assigned post-hoc by a statistical inference algorithm.

The disambiguation algorithm first defines a partition of the set of reified set referents into sets $\{s, s', s''\}$ of reified set referents s whose discourse referent variables ($\mathbf{f_1}\ s$) are connected by semantic dependencies. For example, $\mathbf{s_L}$, $\mathbf{s_C}$ and $\mathbf{s'_N}$ in Figure 4 are part of the same partition, but $\mathbf{s'_L}$ is not.

Scope dependencies are then constructed from these partitions using a greedy algorithm which starts with an arbitrary set from this partition in

[5] Like any other dependency, a continuation dependency may be stored during incremental processing when both its cue (source) and target (destination) referents have been hypothesized. For example, upon processing the word *numbers* in the sentence *Every line contains two numbers,* a continuation dependency may be stored from the nuclear scope set associated with this word to the nuclear scope set of the subject *every line*, forming an in-situ interpretation with some amount of activation (see Figure 4), and with some (probably smaller) amount of activation, a continuation dependency may be stored from the nuclear scope set of this subject to the nuclear scope set of this word, forming an inverted interpretation. See Schuler (2014) for a model of how sentence processing in associative memory might incrementally store dependencies like these as cued associations.

the dependency graph, then begins connecting it, selecting the highest-ranked referent of that partition that is not yet attached and designating it as the new highest-scoping referent in that partition, attaching it as the context of the previously highest-scoping referent in that partition if one exists. This proceeds until:

1. the algorithm reaches a restrictor or nuclear scope referent with a sibling (superset or subset) nuclear scope or restrictor referent that has not yet served as the highest-scoping referent in its partition, at which point the algorithm switches to the partition of that sibling referent and begins connecting that; or

2. the algorithm reaches a restrictor or nuclear scope referent with a sibling nuclear scope or restrictor referent that *is* the highest-scoping referent in its partition, in which case it connects it to its sibling with a continuation dependency from the nuclear scope referent to the restrictor referent and merges the two siblings' partitions.

In this manner, all set referents in the dependency graph are eventually assembled into a single tree of continuation dependencies.

6 Evaluation

This paper defines a graphical semantic representation with desirable properties for storing sentence meanings as cued associations in associative memory. In order to determine whether this representation of continuation dependencies is reliably learnable, the set of test sentences from the QuanText corpus (Manshadi et al., 2011) was automatically annotated with these continuation dependencies and evaluated against the associated set of gold-standard quantifier scopes. The sentences in this corpus were collected as descriptions of text editing tasks using unix tools like sed and awk, collected from online tutorials and from graduate students asked to write and describe example scripts. Gold-standard scoping relations in this corpus are specified over bracketed sequences of words in each sentence. For example, the sentence *Print every line that starts with a number* might be annotated:

Print [$_1$ *every line*] *that starts with* [$_2$ *a number*] .

scoping relations: $1 > 2$

meaning that the quantifier over *lines*, referenced in constituent 1, outscopes the quantifier over *numbers*, referenced in constituent 2. In order to isolate the learnablility of the continuation dependencies described in this paper, both training and test sentences of this corpus were annotated with hand-corrected GCG derivations which are then used to obtain semantic dependencies as described in Section 4. Continuation dependencies are then inferred from these semantic dependencies using the algorithm described in Section 5. Gold-standard scoping relations are considered successfully recalled if a restrictor ($\mathbf{f_1}$ ($\mathbf{f_1}$ i)) or nuclear scope ($\mathbf{f_2}$ ($\mathbf{f_1}$ i)) referent of any lexical item i within the outscoped span is connected by a sequence of continuation dependencies (in the appropriate direction) to any restrictor or nuclear scope referent of any lexical item within the outscoping span.

First, the algorithm was run without any lexicalization on the 94 non-duplicate sentences of the QuanText test set. Results of this evaluation are shown in the third line of Table 1 using the per-sentence complete recall accuracy ('AR') defined by Manshadi et al. (2013).

The algorithm was then run using bilexical weights based on the frequencies $\tilde{\mathsf{F}}(h, h')$ with which a word h' occurs as a head of a category outscoped by a category headed by word h in the 350-sentence training set of the QuanText corpus. For example, since quantifiers over *lines* are often outscoped by quantifiers over *files* in the training data, the system learns to rank continuation dependencies to referents associated with the word *lines* ahead of continuation dependencies to referents associated with the word *files* in bottom-up inference. These lexical features may be particularly helpful because continuation dependencies are generated only between directly adjacent sets. Results for scope disambiguation using these rankings are shown in the fourth line of Table 1. This increase is statistically significant ($p = 0.001$ by two-tailed McNemar's test). This significance for local head-word features on continuation dependencies shows that these dependencies can be reliably learned from training examples, and suggests that continuation dependencies may be a natural representation for scope information.

Interestingly, effects of lexical features for quantifiers (the word *each*, or definite/indefinite distinctions) were not substantial or statistically significant, despite the relatively high frequencies

System	AR
Manshadi and Allen (2011) baseline	63%
Manshadi et al. (2013)	72%
This system, w/o lexicalized model	61%
This system, w. lexicalized model	72%

Table 1: Per-sentence complete recall accuracy ('AR') of tree-based algorithm as compared to Manshadi and Allen (2011) and Manshadi et al. (2013) on explicit NP chunks in the QuanText test set, correcting for use of gold standard trees as described in footnote 19 of Manshadi et al. (2013).

of the words *each* and *the* in the test corpus (occurring in 16% and 68% of test sentences, respectively), which suggests that these words may often be redundant with syntactic and head-word constraints. Results using preferences that rank referents quantified by the word *each* after other referents achieve a numerical increase in accuracy over a model with no preferences (up 5 points, to 66%), but it is not statistically significant ($p = .13$). Results using preferences that rank referents quantified by the word *the* after other referents achieve a numerical increase in accuracy over a model with no preferences (up 1 point, to 62%), but this is even less significant ($p = 1$). Results are even weaker in combination with head-word features (up 1 point, to 73%, for *each*; down two points, to 70%, for *the*). This suggests that world knowledge (in the form of head-word information) may be more salient to quantifier scope disambiguation than many intuitive linguistic preferences.

7 Conclusion

This paper has presented a graphical semantic dependency representation based on bottom-up continuation dependencies which can be translated into lambda calculus, but has the important property that its vertices define a usable set of discourse referents in working memory even in contexts involving conjunction in the scope of quantifiers. An evaluation on an existing quantifier scope disambiguation task shows that non-local continuation dependencies can be as reliably learned from annotated data as representations used in a state-of-the-art quantifier scope resolver. This suggests that continuation dependencies may be a natural representation for scope information.

Continuation dependencies as defined in this paper provide a local representation for quantificational context. This ensures that graphical representations match only when their quantificational contexts match. When used to guide a statistical or vectorial representation, it is possible that this local context will allow certain types of inference to be defined by simple pattern matching, which could be implemented in existing working memory models. Future work will explore the use of this graph-based semantic representation as a basis for vectorial semantics in a cognitive model of inference during sentence processing.

8 Acknowledgements

The authors would like to thank Mehdi Manshadi for assistance in obtaining the QuanText corpus. The authors would also like to thank Erhard Hinrichs, Craige Roberts, the members of the OSU LLIC Reading Group, and the three anonymous *SEM reviewers for their helpful comments about this work.

References

James A. Anderson, Jack W. Silverstein, Stephen A. Ritz, and Randall S. Jones. 1977. Distinctive features, categorical perception and probability learning: Some applications of a neural model. *Psychological Review*, 84:413–451.

Emmon Bach. 1981. Discontinuous constituents in generalized categorial grammars. *Proceedings of the Annual Meeting of the Northeast Linguistic Society (NELS)*, 11:1–12.

Jason Baldridge and Geert-Jan M. Kruijff. 2002. Coupling CCG and hybrid logic dependency semantics. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, Philadelphia, Pennsylvania.

Chris Barker. 2002. Continuations and the nature of quantification. *Natural Language Semantics*, 10:211–242.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4.

Johan Bos. 1996. Predicate logic unplugged. In *Proceedings of the 10th Amsterdam Colloquium*, pages 133–143.

Sarah Brown-Schmidt, Ellen Campana, and Michael K. Tanenhaus. 2002. Reference resolution in the wild: Online circumscription of referential domains in a natural interactive problem-solving task. In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*, pages 148–153, Fairfax, VA, August.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, pages 281–332.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC 2006*.

Judith Degen and Michael K. Tanenhaus. 2011. Making inferences: The case of scalar implicature processing. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, pages 3299–3304.

Marc W. Howard and Michael J. Kahana. 2002. A distributed representation of temporal context. *Journal of Mathematical Psychology*, 45:269–299.

Hans Kamp. 1981. A theory of truth and semantic representation. In Jeroen A. G. Groenendijk, Theo M. V. Janssen, and Martin B. J. Stokhof, editors, *Formal Methods in the Study of Language: Mathematical Centre Tracts 135*, pages 277–322. Mathematical Center, Amsterdam.

Walter Kintsch. 1988. The role of knowledge in discourse comprehension: A construction-integration model. *Psychological review*, 95(2):163–182.

Alexander Koller. 2004. *Constraint-based and graph-based resolution of ambiguities in natural language*. Ph.D. thesis, Universität des Saarlandes.

Geert-Jan M. Kruijff. 2001. *A Categorial-Modal Architecture of Informativity: Dependency Grammar Logic and Information Structure*. Ph.D. thesis, Charles University.

Bill MacCartney and Christopher D. Manning. 2009. An Extended Model of Natural Logic. In *Proceedings of the Eighth International Conference on Computational Semantics*, IWCS-8 '09, pages 140–156. Association for Computational Linguistics.

Mehdi Manshadi and James F. Allen. 2011. Unrestricted quantifier scope disambiguation. In *Graph-based Methods for Natural Language Processing*, pages 51–59.

Mehdi Manshadi, James F. Allen, and Mary Swift. 2011. A corpus of scope-disambiguated english text. In *Proceedings of ACL*, pages 141–146.

Mehdi Manshadi, Daniel Gildea, and James F. Allen. 2013. Plurality, negation, and quantification: Towards comprehensive quantifier scope disambiguation. In *Proceedings of ACL*, pages 64–72.

David Marr. 1971. Simple memory: A theory for archicortex. *Philosophical Transactions of the Royal Society (London) B*, 262:23–81.

David Marr. 1982. *Vision. A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman and Company.

J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102:419–457.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In J. Hintikka, J.M.E. Moravcsik, and P. Suppes, editors, *Approaches to Natural Langauge*, pages 221–242. D. Riedel, Dordrecht. Reprinted in R. H. Thomason ed., *Formal Philosophy*, Yale University Press, 1994.

B.B. Murdock. 1982. A theory for the storage and retrieval of item and associative information. *Psychological Review*, 89:609–626.

Luan Nguyen, Marten van Schijndel, and William Schuler. 2012. Accurate unbounded dependency recovery using generalized categorial grammars. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING '12)*, pages 2125–2140, Mumbai, India.

Richard T. Oehrle. 1994. Term-labeled categorial type systems. *Linguistics and Philosophy*, 17(6):633–678.

Terence Parsons. 1990. *Events in the Semantics of English*. MIT Press.

William Schuler. 2014. Sentence processing in a vectorial model of working memory. In *Fifth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2014)*.

Chung-chieh Shan and Chris Barker. 2006. Explaining crossover and superiority as left-to-right evaluation. *Linguistics and Philosophy*, 29:91–134.

Mark Steedman. 2000. *The syntactic process*. MIT Press/Bradford Books, Cambridge, MA.

Mark Steedman. 2012. *Taking Scope - The Natural Semantics of Quantifiers*. MIT Press.

Michael K. Tanenhaus, Michael J. Spivey-Knowlton, Kathy M. Eberhard, and Julie E. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.

Marten van Schijndel and William Schuler. 2013. An analysis of frequency- and recency-based processing costs. In *Proceedings of NAACL-HLT 2013*. Association for Computational Linguistics.

Marten van Schijndel, Luan Nguyen, and William Schuler. 2013. An analysis of memory-based processing costs using incremental deep syntactic dependency parsing. In *Proceedings of CMCL 2013*. Association for Computational Linguistics.