Following models of distributed associative memory from computational cognitive neuroscience (Marr, 1971; Anderson et al., 1977; Murdock, 1982; McClelland et al., 1995; Howard and Kahana, 2002), the broad-coverage sentence processing model used in this article is defined in terms of referential states, which generalize stimuli as characteristic patterns of neural activation in the brain, and cued associations, which associate referential states through potentiation of synapses between neurons that are active in a cue state and neurons that are active in a target state. In this article, referential states are notated with variables $x$, $y$, and $z$, and cued associations are notated as functions $f$ from (cue) referential states to (target) referential states. Some referential states are then assumed to be elementary predications (Copestake et al., 2005). Elementary predications are referential states which have:

1. *predicate types*, characteristic parts of activation patterns shared across elementary predication instances, notated here by $f_0$ functions from (predication) referential states to type specifications, and

2. distinguished cued associations to *participant* referential states, notated here by numbered functions $f_1$, $f_2$, etc., from (predication) referential states to (participant) referential states.\(^1\)

Collections of referential states connected by elementary predications form cued association structures, notated here using functions $p$ and $q$ from referential states to truth values, which are defined to be true if a particular structure holds at a particular referential state. These cued association structures are similar to semantic dependency structures (Kintsch, 1988; Mel’čuk, 1988; Kruijff, 2001; Baldridge and Kruijff, 2002; Copestake et al., 2005; White, 2006). For example, the cued association structure $p = \lambda z \exists e (f_0 e) = \text{BeingOpen} \land (f_1 e) = z$ defines a structure at a referential state $x$ that is the first participant of a ‘being open’ elementary predication $e$.

The sentence processing model used in this article also assumes referential states that represent narrower generalizations can inherit from referential states

\(^1\)Reciprocal cued associations from participants to elementary predications may also be assumed, but the stronger direction, from elementary predications to unique participants, is notated.
that represent broader generalizations through the use of cued associations distinguished for restriction inheritance, conjunction inheritance and extraction inheritance, notated here as \( f_{\text{rin}} \), \( f_{\text{cin}} \) and \( f_{\text{ein}} \) functions from (narrower) referential states to (broader) referential states. For example, the cued association structure:

\[
\lambda_y \exists x, e, e' \quad (f_0 e) = \text{BeingADoor} \land (f_1 e) = x \land (f_0 e') = \text{BeingOpen} \land (f_1 e') = y \land (f_{\text{rin}} y) = x
\]

defines a dependency structure at a referential state \( y \) that is the first participant of a ‘being open’ elementary predication \( e' \) and inherits from a referential state \( x \) the property of being the first participant of a ‘being a door’ elementary predication \( e \). This inheritance may be used to distinguish argument constraints from modifier constraints, and to distinguish restrictor and nuclear scope arguments of generalized quantifiers, which allows cued association structures to be compiled into a logical form of expressions in typed lambda calculus (Schuler and Wheeler, 2014).

The sentence processing model described in this article operates on referential states for signs (de Saussure, 1916), which are elementary predications connected to signified referential states by cued associations distinguished for signification, notated here as \( f_{\text{sig}} \) functions from (sign) referential states to (signified) referential states. Predication types for these signs, here notated with variables \( \alpha, \beta, \gamma, \delta, \) and \( \varepsilon \) over domain \( S \), may each contain a primitive clausal type \( \tau \) or \( \upsilon \) over domain \( T \) requiring zero or more syntactic arguments \( \varphi \) or \( \psi \) over domain \( O \times S \), where each such argument may have a type-constructing operator (e.g. argument, modifier, conjunct, gap filler) in domain \( O \) followed by a sign type for the argument in domain \( S \). A broad-coverage set of primitive clausal types and type-constructing operators for English is shown in Table 1.

The model described in this article assumes that cued association structures made of elementary predications are composed, stored and retrieved in associative memory according to operations of a left-corner parser (Aho and Ullman, 1972; Johnson-Laird, 1983; Abney and Johnson, 1991; Gibson, 1991; Resnik, 1992; Stabler, 1994; Lewis and Vaisishth, 2005; van Schijndel et al., 2013) using a specific set of semantic processing functions \( R \). These left-corner parser operations process sequences of observed word tokens of type \( \omega, \omega', \omega'' \), etc., by incrementally incorporating them into a cued association structure \( g \). When adjacent words are not directly associated with each other, these cued association structures may consist of one or more sign fragments \( \alpha/\beta \), each a sign of type \( \alpha \) lacking a sign of type \( \beta \) yet to come. For example, a sentence beginning with the words ‘the very,’ may consist of a noun phrase lacking a common noun yet to come (for ‘the’), fol-
Table 1: Primitive clausal types and type-constructing operators for English, adapted from Nguyen et al. (2012).

<table>
<thead>
<tr>
<th>Primitive Clausal Type</th>
<th>Type-Constructing Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>V finite verb</td>
<td>-a preceding argument</td>
</tr>
<tr>
<td>I infinitive</td>
<td>-b following argument</td>
</tr>
<tr>
<td>B base form</td>
<td>-c preceding conjunct</td>
</tr>
<tr>
<td>L participial</td>
<td>-d following conjunct</td>
</tr>
<tr>
<td>A predicative</td>
<td>-g gap filler</td>
</tr>
<tr>
<td>R adverbial</td>
<td>-h heavy shift / extraposition</td>
</tr>
<tr>
<td>G gerund</td>
<td>-i interrogative pronoun</td>
</tr>
<tr>
<td>P particle</td>
<td>-r relative pronoun</td>
</tr>
<tr>
<td>O non-possessive genitive</td>
<td>-v passive</td>
</tr>
</tbody>
</table>

Allowed by an adjective lacking an adjective yet to come (for ‘very’). Cued association structures that consist of multiple sign fragments can be represented as functions with arguments for the holes between these fragments. For example, a cued association structure with holes $h', h$ between sign fragments $\alpha''/\beta'', \alpha'/\beta', \alpha/\beta$ can be represented as a function of type $\beta \rightarrow (\alpha \rightarrow \beta') \rightarrow (\alpha' \rightarrow \beta'') \rightarrow \alpha''$.

Sentence processing in this model starts with a top-level cued association structure (a function from syntactic type $T$ to syntactic type $T$), followed by a sequence of word token units with types $\omega, \omega', \omega''$:

$$\lambda_p : T \lambda_\chi ('p x') : T \cdot \text{unit} : \omega \cdot \text{unit} : \omega' \cdot \text{unit} : \omega'' \cdots,$$

and proceeds by forking off and joining up sign fragments within this structure. At each word $w$, the sentence processing model considers whether to use that word to fork off a new complete sign fragment, using procedurally-learned lexical inference rules $r$ to integrate semantic constraints from the word into the cued association structure. It may decide to fork, creating a new sign of type $\delta$ with a hole between $\delta$ and the bottom of the previous sign fragment $\beta$:

$$\frac{g : \beta \rightarrow \Gamma \cdot w : \omega}{(rgw) : (\delta \rightarrow \beta) \rightarrow \Delta} \quad r : (\beta \rightarrow \Gamma) \rightarrow \omega \rightarrow (\delta \rightarrow \beta) \rightarrow \Delta \in R, \quad (+F)$$

The sentence processing model may also decide not to fork, instead attaching word $w$ at the bottom of the preceding sign fragment, using an identity function ($\lambda_p p$) to fill in the hole between this new complete sign and the bottom of the
previous sign:
\[
\begin{align*}
g : \beta \rightarrow \Gamma & \quad w : \omega \\
(r g w (\lambda_p \ p)) : \Delta & \quad r : (\beta \rightarrow \Gamma) \rightarrow \omega \rightarrow (\delta \rightarrow \beta) \rightarrow \Delta \in R,
\end{align*}
\]  

Lexical inference rules integrate semantic constraints from words of type \( \omega \) into cued association structures \( g \). For example, a lexical rule for a word of type \text{open} would define an elementary predication of type \text{BeingOpen} as the signified referential state of a sign \( x \), with the first participant of that elementary predication as the first participant of \( x \):

\[
\begin{align*}
\lambda_g : \beta \rightarrow \Gamma & \quad \lambda_w : \text{open} \\
(\lambda q h (\lambda_x (f_0 f_{\text{rin}} f_{\text{sig}} x)) = \text{BeingOpen}, (f_1 f_{\text{rin}} f_{\text{sig}} x) = (f_1 x)) : \Gamma \in R
\end{align*}
\]

Each lexical inference rule requires the formation of only a small number of cued associations.

After each fork decision has been made, the sentence processing model considers whether to connect the complete sign fragment of type \( \delta \) resulting from the previous fork decision to the bottom \( \beta \) of the previous disjoint incomplete sign fragment, using procedurally-learned grammatical inference rules \( r \) to compose left children of type \( \delta \) and right children of type \( \epsilon \) into parents of type \( \gamma \). It may decide to join, using an identity function \( (\lambda_p \ p) \) to fill in the hole between the new parent \( \gamma \) and the bottom of the previous sign:

\[
\begin{align*}
g : (\delta \rightarrow \beta) \rightarrow \Gamma & \quad \lambda_q : \epsilon \rightarrow (\gamma \rightarrow \beta) \rightarrow \Delta \in R,
\end{align*}
\]  

The sentence processing model may also decide not to join, instead maintaining a separate sign fragment of type \( \gamma/\epsilon \) with a hole between \( \gamma \) and the bottom \( \beta \) of the previous sign fragment:

\[
\begin{align*}
g : (\delta \rightarrow \beta) \rightarrow \Gamma & \quad \lambda_q : \epsilon \rightarrow (\gamma \rightarrow \beta) \rightarrow \Delta \in R,
\end{align*}
\]  

Finally, the sentence processing model can remove a non-local dependency which no longer appears in any sign fragment following it:

\[
\begin{align*}
g : (\alpha \rightarrow \beta) \rightarrow (\alpha' \rightarrow \beta') \rightarrow (\alpha'' \rightarrow \beta'') \rightarrow \ldots \rightarrow \psi \rightarrow \Gamma & \quad (g h h' h'' \ldots z) : \Gamma \rightarrow \psi \notin \alpha, \beta, \alpha', \beta', \ldots
\end{align*}
\]

A broad-coverage set of grammatical inference rules for English is shown in Tables 2 and 3. Each rule requires the formation of only a small number of cued associations. In the model described in this article, cued associations to older referential states incur integration cost as defined in the DLT.
Table 2: Binary broad-coverage grammatical inference rules in R for argument attachment (Aa, Ab), modifier attachment (Ma, Mb), auxiliary attachment (Ua, Ub), conjunct attachment (Ca, Cb, Cc), gap filler attachment (G), extraposition or heavy shift attachment (H), interrogative pronoun antecedent attachment (I), and relative pronoun antecedent attachment (R), adapted from Nguyen et al. (2012).
These unary rules are combined with other lexical and grammatical inference rules (V), and zero-head introduction (Za, Zb), adapted from Nguyen et al. (2012). Table 3: Unary broad-coverage grammatical inference rules in \( \lambda \) for non-local extraction (Ea, Eb), subject-auxiliary inversion (Q), type-changing (T), passive voice (V), and zero-head introduction (Za, Zb), adapted from Nguyen et al. (2012). These unary rules are combined with other lexical and grammatical inference rules using the recurrences in Equation 1 and 2.
References


