Cued-Association Sentence Processing Markup

June 2, 2018

Contents

1 Overview

This report describes a markup for annotating how sentences are decoded into structures of cued associations, which represent the meanings of these sentences in a distributed associative memory. These decodings may be used to train sentence comprehension models (decoders) and production models (encoders), and to evaluate the linguistic accuracy of these models. Models trained on large corpora of these decodings can then be used to make predictions about effects of storage and retrieval of these associations during human sentence processing.

2 Cued association semantics

Cued associations are rapidly mutable but durable connections between referential states in a distributed associative memory model (Marr, 1971; Anderson et al., 1977; Murdock, 1982; Smolensky, 1990; McClelland et al., 1995; Howard and Kahana, 2002). In a distributed associative memory model, **referential states** x, y are patterns of activation among (primarily cortical) neurons. Cued associations are formed by long-term potentiation at synapses (primarily in the hippocampus) between neurons active in a cue state and neurons active in a target state immediately following the cue state. Outer product models of cued association formation produce human-like effects of interference and superposition. A small set of **association labels** can be distinguished using part of the cue as a label. Labeled associations are then defined over joint features, each of which consists of a pair of one element from the label portion of the pattern and one element from the rest of the cue (Rasmussen and Schuler, 2017).

Graphical structures composed of labeled cued associations can be used to represent decoded meanings of sentences. In this representation, referential states become activated in the presence of certain kinds of stimuli in attentional focus, and thus generalize over those stimuli. Referential states in cued-association structures may therefore be constrained by these structures to respond only to, and thus serve as variables over, particular sets of abstract **discourse entities** u, v from a universe U (Karttunen, 1976), which may persist across encounters. Some referential states may be constrained to respond to **elementary predications** (Copestake et al., 2005) such as eventualities (Davidson, 1967; Parsons, 1990), which are temporally localized, or generalized quantifiers (Barwise and Cooper, 1981), which compare quantities of discourse entities. Some referential states may be constrained to respond to abstract concepts or **predication type constants**. Each referential state that varies over elementary predications is connected by distinctly labeled cued associations to a set of other referential states, which serve as **participants** for that elementary predication. Complex structures composed of several referential states connected by cued associations can be matched to observed or remembered stimuli by shifting attentional focus from referential state to referential state and measuring the similarity of patterns at each predication type constant. In this way, cued-association structures can form descriptions of stimuli of arbitrary complexity.

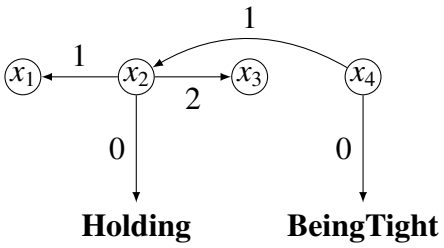


Figure 1: A cued-association structure for the words *hold tightly*.

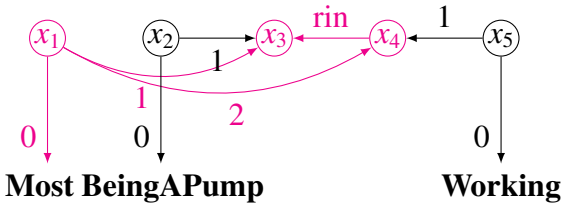


Figure 2: A cued-association structure for the sentence *Most pumps work*.

Cued associations can be represented graphically as labeled directed edges between vertices corresponding to referential states. This report will follow the Mel'čuk (1988) convention of labeling participants with numbers corresponding to the positions of corresponding arguments in a declarative clause, and will additionally label cued associations to predication type constants as '0'. This report will also follow the Parsons (1990) convention of adopting gerund forms for predication type concepts. An example cued-association structure for the words *hold tightly* is shown in Figure 1.

Cued-association structures can also be represented as logical expressions, using numbered functions $\mathbf{f}_n$ from referential states to referential states to define labeled cued associations. For example, the cued-association structure depicted in Figure 1 can be expressed logically as:

$$(\mathbf{f_0}\ x_2)=\mathbf{Holding} \wedge (\mathbf{f_1}\ x_2)=x_1 \wedge (\mathbf{f_2}\ x_2)=x_3 \wedge (\mathbf{f_0}\ x_4)=\mathbf{BeingTight} \wedge (\mathbf{f_1}\ x_4)=x_2 \quad (1)$$

This representation can be simplified using the predication type as a function applied to arguments consisting of the elementary predication and its participants, ordered by cued association labels:

$$(\mathbf{Holding}\ x_2\ x_1\ x_3) \wedge (\mathbf{BeingTight}\ x_4\ x_2) \quad (2)$$

Cued-association structures that involve general knowledge often require **quantifiers**. Most quantifiers take the form of a function over a **restrictor** set and a **nuclear scope** set of discourse referents, which compares the cardinality of the restrictor set to the cardinality of the intersection of the restrictor and nuclear scope sets (Barwise and Cooper, 1981). These generalized quantifiers can be implemented using an elementary predication over two referential states, using the sets of discourse referents that each state responds to. The intersection can be implemented with a **restriction inheritance** association (notated 'rin') from the referential state of the nuclear scope to the referential state of the restriction. This inheritance adds the constraints of the restriction

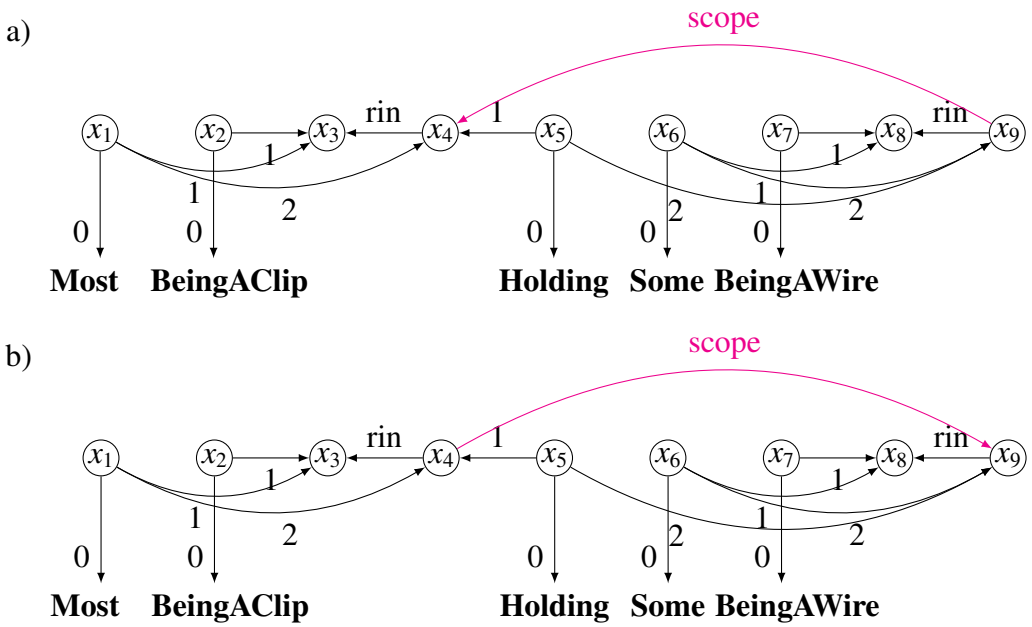


Figure 3: In-situ (a) and inverse (b) scopings of quantifiers in the sentence *Most clips hold a wire*.

to those of the nuclear scope. A cued-association structure for the sentence *Most pumps work,* exemplifying a generalized quantifier, is shown in Figure 2.

Cued-association structures containing multiple quantifiers need some specification of scope. Following Schuler and Wheeler (2014), scope can be specified by introducing a scope association from the nuclear scope of each outsoped quantifier to the nuclear scope of its immediately outscoping quantifier. Two possible scopings of quantifiers in the sentence *Most clips hold a wire,* are shown in Figure 3.

Well-formed scoped and quantified cued-association structures define lambda calculus expressions. For example, the in-situ (a) scoping depicted in Figure 3 can be translated into the lambda calculus expression:

$$\textbf{Most}\ (\lambda_{u_3}\ \textbf{BeingAClip}\ u_3)\ (\lambda_{u_4}\ \textbf{Some}\ (\lambda_{u_8}\ \textbf{BeingAWire}\ u_8)\ (\lambda_{u_9}\ \textbf{Holding}\ u_4\ u_9)) \qquad (3)$$

Rules for translating cued association structures into lambda calculus expressions are provided in Appendix A.

3 Signs and inference rules for sentence processing

Sentences may be produced from and comprehended into cued-association structures using a set of **lexical inference rules** and a set of **grammatical inference rules**. These lexical and grammatical inference rules compositionally associate semantic cued-association structures with recursively nested structures of **signs** (Saussure, 1916), each with associated categories or **sign types** (linked by associations labeled '0') and **signified** semantic cued-association structures (linked by associations labeled 'sig'). In addition to associating semantic structures with word sequences, these rules may also be associated with probabilities, which define preferences in cases of ambiguity, and which

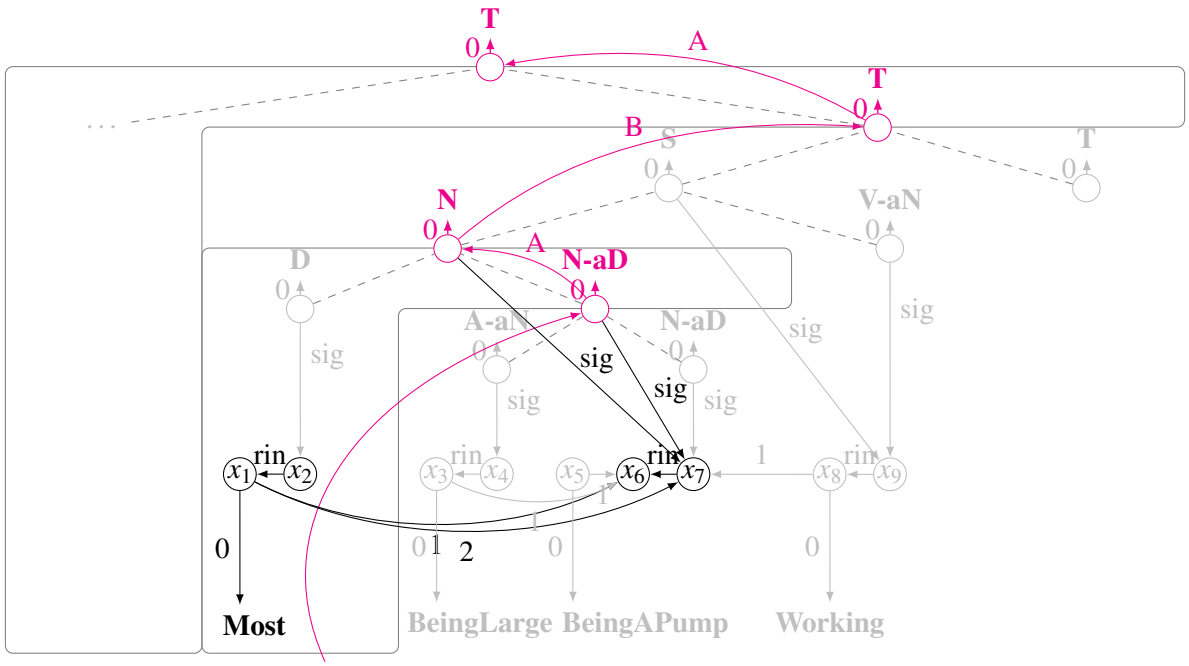


Figure 4: Derivation fragments stored at the word *Most* in the sentence *Most large pumps work*. Cued associations of the store itself are highlighted, accessible signified cued associations are shown in black, and unaccessible or not yet constructed associations are shown in light gray.

can be used to define prefix probabilities and estimates of information-theoretic surprisal (Hale, 2001; Levy, 2008). These rule probabilities may also be used to define gradient grammaticality (Featherston, 2005; Crocker and Keller, 2005), allowing inference rules to overgenerate.

Following a common assumption (Abney and Johnson, 1991; Gibson, 1991; Stabler, 1994; Lewis and Vasishth, 2005; van Schijndel et al., 2013) that sentence processing uses a left-corner parsing strategy (Rosenkrantz and Lewis, 1970; Johnson-Laird, 1983; Resnik, 1992), the lexical and grammatical inference rules in a sentence processing model may be defined over a **pushdown store** that is constrained to hold a limited number of disjoint nested **derivation fragments**, each consisting of an **apex sign** a lacking a **base sign** b yet to come. This kind of store can be defined in a distributed associative memory (Rasmussen and Schuler, 2017) using cued associations from the base to the apex of each derivation fragment (labeled 'A') and from the apex of each derivation fragment to the base of its immediately preceding superordinate derivation fragment (labeled 'B'). This store can also be extended to hold zero or more **non-local dependents**, such as those arising from filler-gap constructions, as additional apices and bases (van Schijndel and Schuler, 2013). Figure 4 shows a set of derivation fragments stored at the word *Most* in the sentence *Most large pumps work*.

The store shown in figure 4 can be notated as the following cued association structure, defined at the base b of the most recent and most subordinate derivation fragment currently in attentional focus:

$$\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{N\text{-}aD} \wedge (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\mathbf{N} \wedge (\mathbf{f_0}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\mathbf{T} \wedge (\mathbf{f_0}{\circ}\mathbf{f_A}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\mathbf{T} \quad (4)$$

This specification can be shortened by translating it into a typed **store function** g:

$$g : \mathbf{N\text{-}aD} \to (\mathbf{N} \to \mathbf{T}) \to \mathbf{T} \tag{5}$$

Here **holes** in the store (e.g. between the sign of type **N** and the lowest sign of type **T** in Figure 4) are represented as functions from cued association structures to cued association structures (e.g. $\mathbf{N} \to \mathbf{T}$). Rules for translating cued association structures into typed store functions are provided in Appendix B.

A left-corner parsing model proceeds by alternately applying lexical inference rules $r \in R_0$ and grammatical inference rules $r \in R_2$ to produce a sequence of store functions. These applications also make decisions about whether to connect derivation fragments at each phase. First, a lexical inference rule can be applied a store that begins with a base, followed by a sequence of input words $w \cdots$, to return a store that begins with an apex, followed by second and subsequent input words:[1]

$$\frac{g : \beta \to \Gamma \;\cdot\; w : \omega \;\cdots}{(r\, g\, w) : (\delta \to \Delta') \to \Gamma' \;\cdots} \; r : \overbrace{(\beta \to \Gamma)}^{\text{old store}} \to \overbrace{\omega}^{\text{word}} \to \overbrace{(\delta \to \Delta') \to \Gamma'}^{\text{new store}} \;\in\; R_0, \tag{L$\downarrow$}$$

$$\frac{g : \beta \to \Gamma \;\cdot\; w : \omega \;\cdots}{(r\, g\, w\, (\lambda_p\, p)) : \Gamma' \;\cdots} \; r : \overbrace{(\beta \to \Gamma)}^{\text{old store}} \to \overbrace{\omega}^{\text{word}} \to \overbrace{(\beta \to \beta) \to \Gamma'}^{\text{new store}} \;\in\; R_0, \tag{L$\uparrow$}$$

Note that the both applications append a hole $\delta \to \Delta'$ or $\beta \to \beta$ onto the store, lengthening it, but the second application matches an identity function $\lambda_p\, p$ of type $\beta \to \beta$ into this hole, closing it and keeping the store the same length. After a lexical inference rule is applied, yielding a store that begins with an apex, followed by a sequence of words, then a binary grammatical inference rule can be applied, yielding a store that begins with a base, followed by that same sequence of words:

$$\frac{g : (\delta \to \Delta) \to \Gamma \;\cdots}{(r\, g) : \varepsilon_{1\ldots n} \to (\gamma \to \Delta') \to \Gamma' \;\cdots} \; r : \overbrace{((\delta \to \Delta) \to \Gamma)}^{\text{old store}} \to \overbrace{\varepsilon_{1\ldots n} \to (\gamma \to \Delta') \to \Gamma'}^{\text{new store}} \;\in\; R_2, \tag{G$\downarrow$}$$

$$\frac{g : (\delta \to \Delta) \to \Gamma \;\cdots}{\lambda_{q_{1\ldots n} : \varepsilon_{1\ldots n}} (r\, g\, q_{1\ldots n}\, (\lambda_p\, p)) : \Gamma' \;\cdots} \; r : \overbrace{((\delta \to \Delta) \to \Gamma)}^{\text{old store}} \to \overbrace{\varepsilon_{1\ldots n} \to (\gamma \to \gamma) \to \Gamma'}^{\text{new store}} \;\in\; R_2, \tag{G$\uparrow$}$$

Note that again, the second application matches an identity function $\lambda_p\, p$ of type $\gamma \to \gamma$ into the first hole of the store, closing the hole and shortening the store.

Lexical and binary grammatical inference rules in R_0 and R_2 may be directly learned as part of the language acquisition process, or may be derived from application of zero or more **unary grammatical inference rules** from R_1:

$$\text{for } \overbrace{r : (\beta \to \Gamma) \to \omega \to (\delta \to \Delta) \to \Gamma'}^{\text{lexical inference rule}} \in R_0, \;\; \overbrace{r' : ((\delta \to \Delta) \to \Gamma') \to (\delta' \to \Delta') \to \Gamma''}^{\text{unary rule}} \in R_1,$$
$$r' \circ r : (\beta \to \Gamma) \to \omega \to (\delta' \to \Delta') \to \Gamma'' \;\in\; R_0 \tag{8a}$$

$$\text{for } \overbrace{r : ((\delta \to \Delta) \to \Gamma) \to \varepsilon_{1..n} \to (\gamma \to \Delta') \to \Gamma'}^{\text{grammatical inference rule}} \in R_2, \;\; \overbrace{r' : ((\gamma \to \Delta') \to \Gamma') \to (\gamma' \to \Delta'') \to \Gamma''}^{\text{unary rule}} \in R_1,$$
$$r' \circ r : ((\delta \to \Delta) \to \Gamma) \to \varepsilon_{1..n} \to (\gamma' \to \Delta'') \to \Gamma'' \;\in\; R_2 \tag{8b}$$

[1] In specifications of inference rules over typed store functions, lowercase Greek letters $\alpha, \beta, \gamma, \delta, \varepsilon$ serve as variables over simple sign types like **N** or **T**, and uppercase Greek letters Γ, Δ serve as variables over functional types of arbitrary complexity.

4 Broad-coverage sign types for English

The model of English sentence processing described in this paper is intended to produce accurate semantic representations for English sentences with a minimum number of inference rules, which are presumed to be learned during early childhood. Since English is a fixed word order language, the grammatical inference rules will resemble those of a categorial grammar (Ajdukiewicz, 1935; Bar-Hillel, 1953; Oehrle, 1994).[2] Sign types or categories in this model are defined recursively as **primitive types** $\tau, \upsilon \in \{\mathbf{S}, \mathbf{N}, \dots\}$ followed by zero or more **dependencies** ϕ, ψ, each of which consists of a **type-combining operator** $o \in \{\textbf{-a}, \textbf{-b}, \dots\}$ followed by another sign type.

4.1 Primitive types

This model assumes the following primitive types, which are types for signs that do not have any unsatisfied dependencies. In most cases these types correspond to clauses. To contrast the forms, clausal types will be exemplified by a sentence with the meaning: [3]

$$(\textbf{BeingHim}\ x_1\ x_2) \wedge (\textbf{Knowing}\ x_3\ x_2\ x_5) \wedge (\textbf{BeingAda}\ x_4\ x_5) \tag{9}$$

Top-level discourse (T). A top level discourse is a sequence of sentences.

(1) $[_{\mathbf{T}}$ She found someone. He knows Ada.$]$

It is used as the apex and base of the most superordinate derivation fragment on the pushdown store during comprehension.

Sentence (S). A sentence is usually a main clause, delimited by punctuation:

(2) She says: $[_{\mathbf{S}}$ Know Ada$]$!

This type subsumes declarative, imperative, open interrogative and closed interrogative clauses, which can be coordinated as a sentence:

(3) $[_{\mathbf{S}}$ Go down the hall$]$ and $[_{\mathbf{S}}$ it's the first door on the left$]$.

Finite verb clause (V). Finite clauses are declarative sentences with nominative subjects which may be subordinate:

(4) She believes that $[_{\mathbf{V}}$ he knows Ada$]$.

This type subsumes both simple present and preterite forms, which can be coordinated:

(5) She believes that $[_{\mathbf{V}}$ he knows Ada$]$ and $[_{\mathbf{V}}$ he knew Ada$]$.

Finite verb clauses are selected by complementizers like *that*.

[2] Specifically, this paper adopts the notation of Nguyen et al. (2012), which is based on the treatment of extraposition and other non-local dependencies in Bach (1981).

[3] These examples use the masculine singular pronoun because it distinguishes nominative, accusative and genitive case.

Infinitive clause (I). Infinitive clauses include an accusative subject followed by an infinitive particle *to* followed by a base-form or bare infinitive phrase:

(6) She allows [$_{\mathbf{I}}$ him to know Ada].

Infinitive clauses are selected by verbs like *allow*. Infinitive phrases are selected by verbs like *want*.

Bare infinitive or base-form clause (B). Bare infinitive or base-form clauses have nominative subjects and are headed by the uninflected or lemma form of a verb:

(7) She requires that [$_{\mathbf{B}}$ he know Ada].

Bare infinitive clauses are selected by verbs like *require*. Bare infinitive phrases are selected by modal auxiliaries and the infinitive particle *to*.

Participial form (L). Participial clauses are not attested in English, but participial phrases are headed by the participial form of a verb:

(8) They have [$_{\mathbf{L\text{-}aN}}$ known Ada]

Participial phrases are selected by verbs like *have*. Although passives use the participial verb form, this clause type does not include passives.

Predicative or small clause (A). Predicative or small clauses have accusative subjects and are headed by an adjective, a preposition, a present participle verb, or a verb passive in passive voice, or a noun:

(9) She kept [$_{\mathbf{A}}$ him knowing Ada].

Predicative clauses are selected by verbs like *consider*, *leave* or *keep*. Predicative phrases serve as pre- and post-modifiers to nouns, and are selected by the copula *be* and *seem*. Predicative uses of noun phrases, for example following a copula or in an appositive modifier, are also considered predicative phrases. This type subsumes adjective clauses, prepositional clauses, present participle clauses, passive clauses and headless copular clauses, all of which can be coordinated:

(10) She considers [$_{\mathbf{A}}$ Kim smart] and [$_{\mathbf{A}}$ Pat a fool].

Predicative clauses also subsume reduced relatives in a post-nominal context.

Adverbial form (R). Adverbial clauses are not attested in English, but adverbial phrases serve as modifiers to verbs, prepositions, adjectives and other adverbs:

(11) They do it [$_{\mathbf{R\text{-}aN}}$ knowingly].

Adverbial phrases also subsume prepositional phrases, temporal noun phrases, and measure noun phrases, all of which can be coordinated:

(12) Open the windows [$_{\mathbf{R\text{-}aN}}$ an inch] or [$_{\mathbf{R\text{-}aN}}$ completely].

Gerund clause (G). Gerund clauses are characterized by subjects in accusative case and are headed by present participial verbs:

(13) She works without $[_{\mathbf{G}}$ him knowing Ada$]$.

Gerund clauses are selected by prepositions like *without*. Gerund phrases are selected by prepositions like *by*.

Particle (P). Particles are semantically empty arguments, usually of verbs:

(14) She picked him $[_{\mathbf{Pup}}$ up$]$.

Particle types always specify a particular word and do not coordinate.

Subject-auxiliary inversion (Q). Subject auxiliary inverted clauses are characterized by auxiliaries followed by nominative subjects:

(15) She asks: $[_{\mathbf{Q}}$ did he know Ada$]$?

These also occur in subordinate clauses, for example following *nor*.

Subordinate finite verb (C). Subordinate finite verb clauses are characterized by a subordinizer or complementizer *that* followed by a finite verb clause:

(16) She fretted $[_{\mathbf{C}}$ that he knows Ada$]$.

This type is selected by verbs like *fret*, which do not select simple finite verb clases. This type also includes variants **Cthan** and **Cas** for complements of comparatives.

Subordinate infinitive (F). Subordinate infinitive clauses are characterized by a subordinizer *for* followed by an infinitive clause:

(17) She waits $[_{\mathbf{F}}$ for him to know Ada$]$.

This type is selected by verbs like *wait*.

Subordinate bare infinitive (E). Subordinate bare infinitive clauses are characterized by a subordinizer or complementizer *that* followed by a bare infinitive clause:

(18) She requires $[_{\mathbf{E}}$ that he know Ada$]$.

This type is selected by verbs like *require*.

Noun phrase or nominal clause (N). A noun phrase is headed by a noun and is characterized by genitive subjects, adjectival modifiers, and post-nominal complements preceded by the grammaticalized preposition *of*:

(19) She talks about $[_{\mathbf{N}}$ his knowledge of Ada$]$.

This category also includes variants **Ne** and **Nc** for expletive instances of *it* used in *it*-extraposition and *it*-clefts, respectively.

Genitive or possessive determiner (D). Genetives consist of possessive nouns or noun phrases followed by a *'s* marker:

(20) She calculates [$_{\mathbf{D}}$ his knowledge of Ada's] effect.

This type does not include ordinary determiners like *a* and *the* which are type-raised as quantifiers of type **N-b(N-aD)** in order to have access to the nuclear scope states of their arguments.

Grammaticalized preposition (O). Grammaticalized prepositions like *of* are markers that have no internal semantics:

(21) She tires [$_{\mathbf{O}}$ of his knowledge of Ada].

This type does not cover instances of *of* that are paraphrasable as *have*, which are instead treated as ungrammaticalized prepositions **A-aN-bN**. This type is also selected by verbs like *despair* or adjectives like *wary*.

4.2 Type-combining operators

The following type combining operators are used to distinguish unsatisfied argument dependencies.

Unsatisfied requirements for preceding and succeeding arguments (-a and -b). Some signs expect other signs as arguments immediately ahead of or behind themselves. These requirements are specified in the sign type:

(22) a. Kim [$_{\mathbf{V\text{-}aN}}$ knows Ada].
b. Kim thinks [$_{\mathbf{C\text{-}bV}}$ that] Pat knows Ada.

Types for signs that lack multiple arguments list these arguments in the order in which they attach, from the bottom up:

(23) Kim [$_{\mathbf{V\text{-}aN\text{-}bN}}$ knows] Ada.

Signs with unsatisfied arguments must be distinguished from satisfied signs because these two sign types do not coordinate:

(24) * Kim thinks that [$_{\mathbf{V}}$ [$_{\mathbf{V\text{-}aN}}$ knows Ada] and [$_{\mathbf{V}}$ Pat knows Lisp]].

Similarly signs that lack arguments ahead do not substitute or coordinate with signs that lack the same argument behind:

(25) * Kim [$_{\mathbf{V\text{-}aN}}$ [$_{\mathbf{V\text{-}bN}}$ Pat knows] and [$_{\mathbf{V\text{-}aN}}$ knows Ada]].

Unsatisfied requirements for preceding and succeeding conjuncts (-c and -d). Unsatisfied dependencies on conjuncts are also distinguished in sign types:

(26) Kim knows Ada [$_{\text{N-cN}}$ and Lisp].

Note: in order to reduce data sparsity, conjunctions with succeeding conjuncts are marked up with underspecified types:

(27) Kim knows Ada [$_{\text{X-cX-dX}}$ and] Lisp.

Signs with unsatisfied conjuncts must be distinguished from other types of signs because signs with unsatisfied conjuncts do not coordinate, even with signs of the same type:

(28) a. * Kim thinks that [$_{\text{V}}$ [$_{\text{V-cN}}$ and Lisp] and [$_{\text{V}}$ Pat knows Ada]].
b. * Kim [$_{\text{V-aN}}$ [$_{\text{V-cN}}$ and Lisp] and [$_{\text{V-aN}}$ knows Ada]].
c. * Kim knows Ada [$_{\text{N-cN}}$ [$_{\text{N-cN}}$ and Lisp] or [$_{\text{N-cN}}$ and Perl]].

Unsatisfied requirements for gap fillers (-g). Signs in filler-gap constructions have requirements for fillers, which need not immediately precede or succeed the sign:

(29) Ada is [$_{\text{N}}$ a language] that [$_{\text{V-gN}}$ Pat thinks _ has objects].

Signs with unsatisfied gap fillers must be distinguished from satisfied signs because these two types of signs do not coordinate:

(30) * Kim thinks that [$_{\text{V}}$ [$_{\text{V-gN}}$ Pat thinks _ has objects] and [$_{\text{V}}$ Ada uses objects]].

Signs with unsatisfied gap fillers also do not substitute or coordinate with signs that have other types of dependencies, and so must be distinguished:

(31) * Ada [$_{\text{V-aN}}$ [$_{\text{V-aN}}$ uses objects] and [$_{\text{V-gN}}$ Pat thinks _ has objects]].

Unsatisfied requirements for heavy-shifted or postposed categories (-h). Some signs have other signs postposed or heavy-shifted away, which need not immediately precede or succeed the sign:

(32) [$_{\text{V-hN}}$ Kim has learned _] this morning [$_{\text{N}}$ the mysterious Ada programming language].

Signs with unsatisfied heavy shifts or postposings must be distinguished from satisfied signs, because these two types of signs do not coordinate:

(33) * [$_{\text{V}}$ [$_{\text{V}}$ Ada has objects] and [$_{\text{V-hN}}$ Kim has learned _]].

Signs with unsatisfied heavy shifts or postposings also do not substitute or coordinate with signs that have other types of dependencies, and so must be distinguished:

(34) * [$_{\text{V-hN}}$ [$_{\text{V-hN}}$ Kim has learned _] and [$_{\text{V-gN}}$ Pat thinks _ has objects]] [$_{\text{N}}$ the mysterious Ada programming language].

Right node raising constructions are also analyzed using this dependency:

(35) [$_{\text{V-hN}}$ [$_{\text{V-hN}}$ Kim has learned _] and [$_{\text{V-hN}}$ Pat has forgotten _]] [$_{\text{N}}$ the mysterious Ada programming language].

Heavy shift and right node raising dependencies are not distinguished, because they seem to occur in the same contexts:

(36) [$_{\text{V-hN}}$ [$_{\text{V-hN}}$ Kim has learned _] and [$_{\text{V-hN}}$ Pat has forgotten _]] this morning [$_{\text{N}}$ the mysterious Ada programming language].

Unsatisfied requirements for passivized arguments (-v). Passives allow a very constrained form of non-local dependency to the subject:

(37) a. Ada is considered [$_{\text{A-vN}}$ _ object-oriented].
b. This bridge has been slept [$_{\text{R-aN-vN}}$ under _].

Signs lacking passivized arguments must be distinguished from satisfied signs because these two types of signs do not coordinate:

(38) * Kim considers [$_{\text{A}}$ [$_{\text{A-vN}}$ _ object-oriented] and [$_{\text{A}}$ Ada structured]].

Passivized arguments also do not substitute or coordinate with signs that have other types of dependencies, and so must be distinguished:

(39) * Ada is considered [$_{\text{A-vN}}$ [$_{\text{A-vN}}$ _ object-oriented] and [$_{\text{A-hN}}$ Kim proficient in _]].

Unsatisfied requirements for interrogative or relative pronoun antecedents (-i or -r). Antecedents of interrogative and relative pronouns can also form syntactically-constrained non-local dependencies:

(40) a. Kim knows a language [$_{\text{C-rN}}$ [$_{\text{N-rN}}$ which] defines classes].
b. Kim knows a language [$_{\text{C-rN}}$ in [$_{\text{N-rN}}$ which] classes are defined].
c. Kim wonders [$_{\text{V-iN}}$ [$_{\text{N-iN}}$ what] Ada is].

Signs lacking interrogative and relative pronoun antecedents must be distinguished from satisfied signs because these two types of signs do not coordinate:

(41) * Kim knows a language [$_{\text{C-rN}}$ [$_{\text{C-rN}}$ which _ defines classes] and [$_{\text{C}}$ that it defines classes]].

Passivized arguments also do not substitute or coordinate with signs that have other types of dependencies, and so must be distinguished:

(42) * Kim knows a language [$_{\text{N-rN}}$ [$_{\text{N-rN}}$ which] and [$_{\text{N-iN}}$ what]] defines classes.

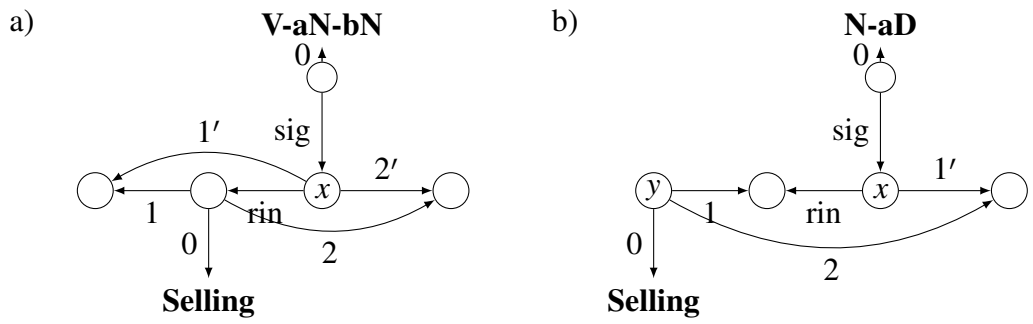


Figure 5: Lexically inferred cued-association structures for the words *sells*, *selling* or *sale*, of types **V-aN-bN**, **A-aN-bN** or **N-aD-bO** respectively (a) and for the word *seller* of type **N-aD** (b).

5 Broad-coverage lexical inference rules for English

Sentence meanings are based on the meanings of words. These meanings can be introduced by lexical inference rules, here defined as functions from stores g that begin with a base sign and words w to stores $\lambda_h \ldots$ that begin with an apex sign (which is the lexical sign for that word).

Lexical inference rules define mappings between the syntactic dependencies of a lexical sign and the participants of the elementary predication which that sign signifies. These mappings are defined through **syntactic argument associations** labeled $1'$, $2'$, $3'$, etc., from each directly signified referential state to the appropriate participant of the signified elementary predication.[4] A lexical sign may refer to an elementary predication itself, as is the case for verbs, adjectives, adverbs and prepositions, or it may refer to a participant of an elementary predication, as is the case for nouns. If the lexical sign refers to a participant, that participant will not be mapped to any of the syntactic argument associations of the sign. Figure 5 shows graphical representations of lexical inference rules for signs that refer to an elementary predication and a participant. These inference rules are also represented by the functions:[5]

$$\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\textbf{sells}}\ \lambda_{h:\textbf{V-aN-bN}\to\beta}\ (g{\circ}h\,(\lambda_x\,(\mathbf{f_0}{\circ}\mathbf{f_{rin}}\,x){=}\textbf{Selling}, (\mathbf{f_1}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{1'}}\,x), (\mathbf{f_2}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{2'}}\,x)))\!:\!\Gamma\ \in\ R_0 \quad (10a)$$

$$\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\textbf{selling}}\ \lambda_{h:\textbf{A-aN-bN}\to\beta}\ (g{\circ}h\,(\lambda_x\,(\mathbf{f_0}{\circ}\mathbf{f_{rin}}\,x){=}\textbf{Selling}, (\mathbf{f_1}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{1'}}\,x), (\mathbf{f_2}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{2'}}\,x)))\!:\!\Gamma\ \in\ R_0 \quad (10b)$$

$$\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\textbf{sale}}\ \lambda_{h:\textbf{N-aD-bO}\to\beta}\ (g{\circ}h\,(\lambda_x\,(\mathbf{f_0}{\circ}\mathbf{f_{rin}}\,x){=}\textbf{Selling}, (\mathbf{f_1}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{1'}}\,x), (\mathbf{f_2}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{2'}}\,x)))\!:\!\Gamma\ \in\ R_0 \quad (10c)$$

$$\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\textbf{seller}}\ \lambda_{h:\textbf{N-aD}\to\beta}\ (g{\circ}h\,(\lambda_x\,\exists_y\,(\mathbf{f_0}\,y){=}\textbf{Selling}, (\mathbf{f_1}\,y){=}(\mathbf{f_{rin}}\,x), (\mathbf{f_2}\,y){=}(\mathbf{f_{1'}}\,x)))\!:\!\Gamma\ \in\ R_0 \quad (10d)$$

Regardless of whether a lexical sign refers to an elementary predication itself or a participant, the referential state that is directly signified by the sign is always a nuclear scope, and the elementary predication or participant is connected to this signified referential state by a restriction inheritance association, as its restrictor. This applies constraints of noun predicates to restrictions in quantified noun phrases, as expected. It also applies constraints of verb predicates to restrictions of adverbial quantifiers, which is also desired. For example, in the sentence *Kim usually likes that*

[4]It may seem strange that syntactic argument associations are specified at signified states rather than signs. This specification was intended to simplify argument propagation in grammatical inference rules.

[5]Note that x and y are variables over referential states, not over the universe of discourse entities, so existential quantifiers e.g. over y indicate the presence of the term in a description rather than the presence of the corresponding entity in the discourse.

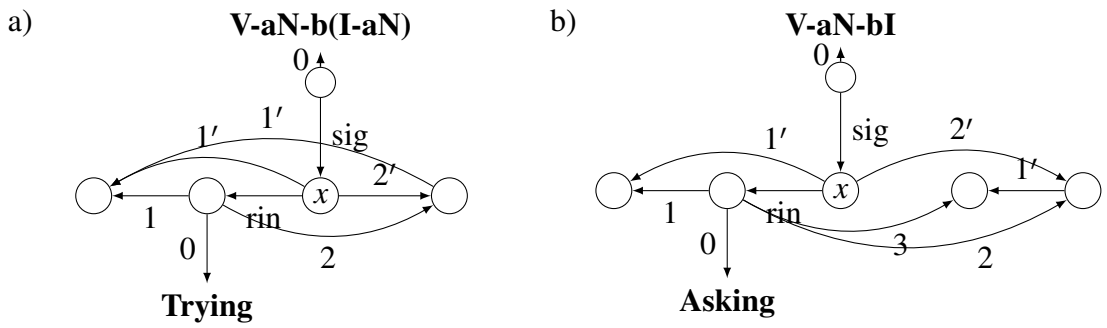


Figure 6: Lexically inferred cued-association structures for the subject control verb *try*, of type **V-aN-b(I-aN)** (a), and for the object control verb *ask*, of type **V-aN-bI** (b).

people know Ada, the quantifier *usually* will be applied to a restrictor set of eventualities of people knowing Ada, and a nuclear scope set of things Kim likes, so the quantifier will be satisfied if most of the knowing eventualities are liked.

Subject control verbs. A subject control verb is a verb that uses its subject as the subject of its phrasal complement. For example, in the sentence *Kim tried to stop, Kim* is both the trier and the stopper. Subject control is modeled in lexical inference rules using syntactic argument associations $(\mathbf{f_{1'}}\,x)=(\mathbf{f_{1'}}\circ\mathbf{f_{2'}}\,x)$ from complements to subjects:

$$\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\mathbf{try}}\ \lambda_{h:\mathbf{V\text{-}aN\text{-}b(I\text{-}aN)}\to\beta}\ (g\circ h\,(\lambda_x\,(\mathbf{f_0}\circ\mathbf{f_{rin}}\,x)=\mathbf{Trying},(\mathbf{f_1}\circ\mathbf{f_{rin}}\,x)=(\mathbf{f_{1'}}\,x),\\ (\mathbf{f_2}\circ\mathbf{f_{rin}}\,x)=(\mathbf{f_{2'}}\,x),(\mathbf{f_{1'}}\,x)=(\mathbf{f_{1'}}\circ\mathbf{f_{2'}}\,x)))\!:\!\Gamma\ \in\ R_0 \qquad (11)$$

Cued-association structures inferred from this rule are shown in Figure 6a.

Object control verbs. An object control verb is a verb that uses its indirect object as the subject of its phrasal complement, or equivalently, uses the subject of its small clause complement as a participant in its elementary predication. For example, in the sentence *Kim asked Pat to stop, Pat* is both the askee and the stopper. Object control is modeled in lexical inference rules using participant associations $(\mathbf{f_3}\circ\mathbf{f_{rin}}\,x)=(\mathbf{f_{1'}}\circ\mathbf{f_{2'}}\,x)$ from elementary predications to subjects of complements:

$$\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\mathbf{ask}}\ \lambda_{h:\mathbf{V\text{-}aN\text{-}bI}\to\beta}\ (g\circ h\,(\lambda_x\,(\mathbf{f_0}\circ\mathbf{f_{rin}}\,x)=\mathbf{Asking},(\mathbf{f_1}\circ\mathbf{f_{rin}}\,x)=(\mathbf{f_{1'}}\,x),\\ (\mathbf{f_2}\circ\mathbf{f_{rin}}\,x)=(\mathbf{f_{2'}}\,x),(\mathbf{f_3}\circ\mathbf{f_{rin}}\,x)=(\mathbf{f_{1'}}\circ\mathbf{f_{2'}}\,x)))\!:\!\Gamma\ \in\ R_0 \qquad (12)$$

Cued-association structures inferred from this rule are shown in Figure 6b.

Auxiliary verbs. Auxiliary verbs are similar to subject control verbs but do not semantically constrain their subjects. For example, in the sentence *Kim was stopping, Kim* is not an participant in the elementary predication of the tense marker.

$$\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\mathbf{was}}\ \lambda_{h:\mathbf{V\text{-}aN\text{-}b(A\text{-}aN)}\to\beta}\ (g\circ h\,(\lambda_x\,(\mathbf{f_0}\circ\mathbf{f_{rin}}\,x)=\mathbf{BeingBeforeNow},\\ (\mathbf{f_1}\circ\mathbf{f_{rin}}\,x)=(\mathbf{f_{2'}}\,x),(\mathbf{f_{1'}}\,x)=(\mathbf{f_{1'}}\circ\mathbf{f_{2'}}\,x)))\!:\!\Gamma\ \in\ R_0 \qquad (13)$$

Cued-association structures inferred from this rule are shown in Figure 7a.

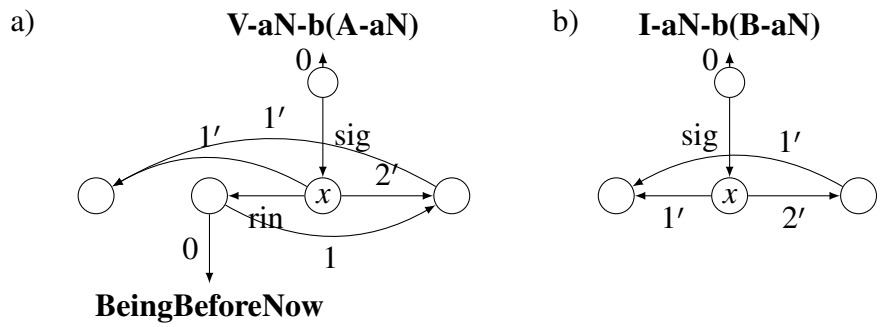


Figure 7: Lexically inferred cued-association structures for the auxiliary verb *was*, of type **V-aN-b(A-aN)** (a), and for the infinitive marker *to*, of type **I-aN-b(B-aN)** (b).

Markers. Markers in general are particles with no lexical meaning, which serve to cast their complements to different types. English uses a finite subordinate marker *that* of type **C-bV** which casts finite verb clauses **V** to subordinate clauses **C**, an infinitive subordinate marker *for* of type **F-bI** which casts infinitive clauses **I** to subordinate infinitive clauses **F**, a genitive marker *'s* of type **D-aN** which casts noun phrases **N** to genitives **D**, and a grammaticalized preposition *of* of type **O-bN** which casts noun phrases **N** to grammaticalized prepositional phrases **O**. Some markers like the infinitive particle *to* also use their subjects as the subjects of their phrasal complements, like subject control or auxiliary verbs:

$$\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\mathbf{was}}\ \lambda_{h:\mathbf{I\text{-}aN\text{-}b(B\text{-}aN)}\to\beta}\ (g{\circ}h\,(\lambda_x\,(\mathbf{f_{1'}}\,x){=}(\mathbf{f_{1'}}{\circ}\mathbf{f_{2'}}\,x)))\!:\!\Gamma\ \in\ R_0 \tag{14}$$

Cued-association structures inferred from this rule are shown in Figure 7b.

Quantifier determiners. A quantifier determiner is a determiner that signifies a quantifier function over a restrictor and nuclear scope set defined by its context. A quantifier determiner signifies an elementary predication whose first and second participants are the restrictor and nuclear scope referential states signified by the noun phrase in which it occurs. This requires that a quantifier determiner take its sibling as a syntactic argument:

$$\begin{aligned}\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\mathbf{most}}\ \lambda_{h:\mathbf{N\text{-}b(N\text{-}aD)}\to\beta}\ (g{\circ}h\,(\lambda_x\,(\mathbf{f_0}{\circ}\mathbf{f_{rin}}\,x){=}\mathbf{Most},\,(\mathbf{f_1}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{rin}}{\circ}\mathbf{f_{1'}}\,x),\\ (\mathbf{f_2}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{1'}}\,x)))\!:\!\Gamma\ \in\ R_0\end{aligned} \tag{15}$$

Cued-association structures inferred from this rule are shown in Figure 8a.

Non-subject possessive genitives. In addition to the semantically empty genitive marker, the particle *'s* can also have a possessive meaning expressible by the predicate **Having**. The first participant of this elementary predication is the sibling of the possessive (the first syntactic argument), and the second participant is the sibling of the determiner (the second syntactic argument):

$$\begin{aligned}\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\mathbf{'s}}\ \lambda_{h:\mathbf{N\text{-}b(N\text{-}aD)\text{-}aN}\to\beta}\ (g{\circ}h\,(\lambda_x\,(\mathbf{f_0}{\circ}\mathbf{f_{rin}}\,x){=}\mathbf{Having},\,(\mathbf{f_1}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{2'}}\,x),\\ (\mathbf{f_2}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{rin}}{\circ}\mathbf{f_{1'}}\,x)))\!:\!\Gamma\ \in\ R_0\end{aligned} \tag{16}$$

Cued-association structures inferred from this rule are shown in Figure 8b.

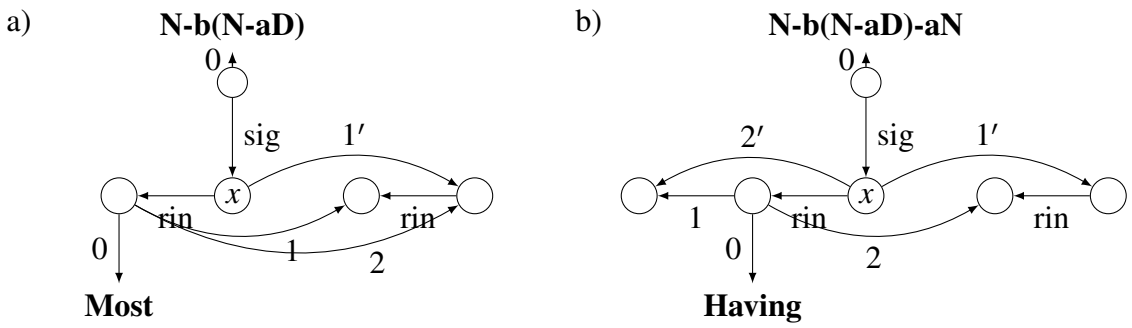


Figure 8: Lexically inferred cued-association structures for the quantifier *most*, of type **N-b(N-aD)** (a), and for the non-subject genitive *'s*, of type **N-b(N-aD)-aN** (b).

6 Broad-coverage grammatical inference rules for English

Languages combine meanings when they combine signs. This composition process can be modeled using grammatical inference rules, defined here as functions from stores g that begin with an apex sign to stores $\lambda_q\,\lambda_h\ldots$ that begin with a base sign.

6.1 Discard (binary)

The simplest way to combine two meanings is to choose one of them as the meaning of the combined structure and discard the other. The discarded cued-association structure may be otherwise connected to the retained structure by anaphoric associations, or it may remain disjoint. For example, parenthetical statements that do not contain anaphoric references to the external context may become disjoint at the conclusion of the parenthetical, and entire connected discourses may become disjoint from the current discourse after a topic shift. Disjoint discourses are still accessible by recall however—for example, they may serve as antecedents of subsequent anaphora.

Grammatical inference rules for discarding preceding and succeeding cued-association structures (Da and Db, respectively) are defined below:

$$\lambda_{g:(\delta\to\Delta)\to\Gamma}\ \lambda_{q:\gamma}\ \lambda_{h:\gamma\to\Delta}\ (g\ \underbrace{(\lambda_p\, h\ \overbrace{(\lambda_y\,\exists_x\,(p\,x),(q\,y))}^{\text{cued-association structure for parent}})}_{\text{function from left child to bottom of previous derivation fragment}}):\Gamma\ \in\ R_2 \tag{Da}$$

$$\lambda_{g:(\gamma\to\Delta)\to\Gamma}\ \lambda_{q:\delta}\ \lambda_{h:\gamma\to\Delta}\ (g\,(\lambda_p\, h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y)))):\Gamma\ \in\ R_2 \tag{Db}$$

Cued associations resulting from these inference rules are shown in Figure 9.

6.2 Argument or complement attachment (binary)

One of the most common ways to combine two signs is to make the meaning of one of them a syntactic argument of the meaning of the other. This kind of attachment is used to connect subjects and complements of verbs, and complements of prepositions, adjectives and nouns. If one sign

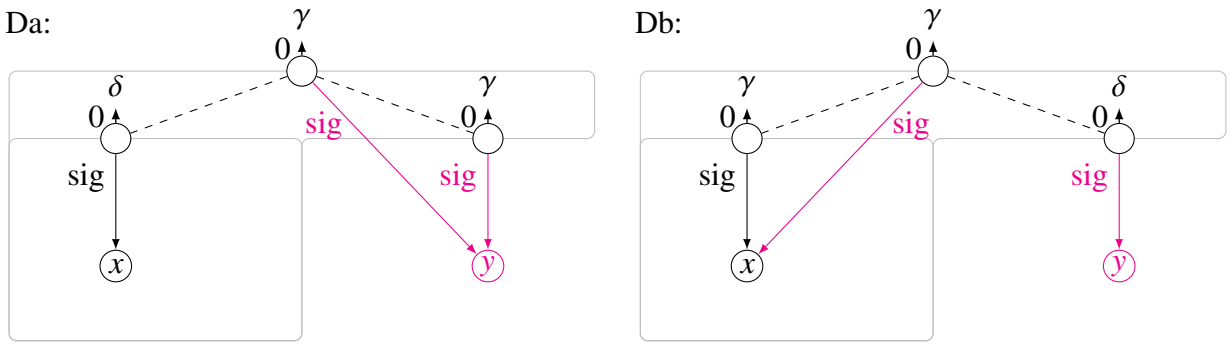


Figure 9: Results of grammatical inference rules for discarding preceding cued-association structures (a) and succeeding cued-association structures (b). Existing structures are shown in black. Structures associated as a result of these rules are highlighted.

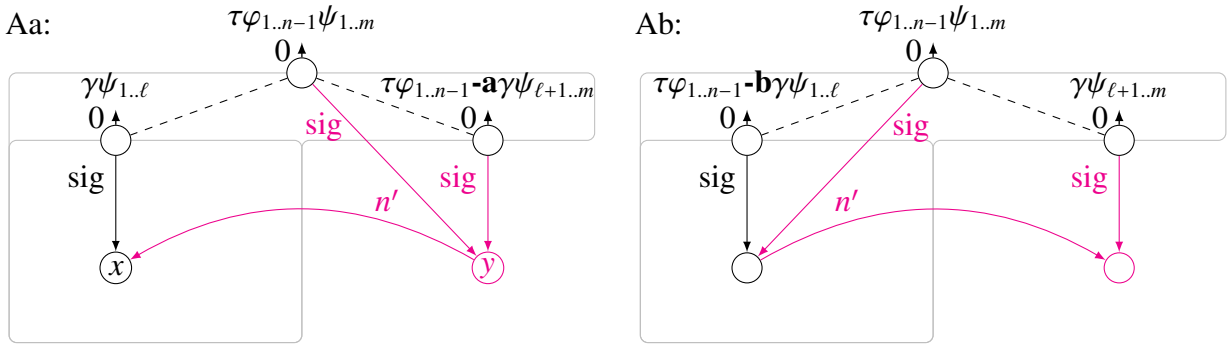


Figure 10: Results of grammatical inference rules for attaching cued-association structures of preceding signs (a) and succeeding signs (b) as arguments. Existing structures are shown in black. Structures associated as a result of these rules are highlighted.

has a type γ and the other has a type consisting of a primitive τ followed by n argument dependencies $\varphi_{1..n}$, the last of which is **-a** or **-b** followed by a γ, then the signified structure of the sign of type γ may be attached to the signified structure of the other sign by a syntactic argument association with label n'. The resulting structure will be of type $\tau\varphi_{1..n-1}$, lacking the final dependency, which is now satisfied. Additionally, any non-local dependencies $\psi_{1..\ell}$ on the first sign and any non-local dependencies $\psi_{\ell+1..m}$ on the second sign will be concatenated and propagated up into the combined sign as $\psi_{1..m}$:

$$\lambda_{g:(\gamma\psi_{1..\ell}\to\Delta)\to\Gamma}\ \lambda_{q:\tau\varphi_{1..n-1}\text{-}\mathbf{a}\gamma\psi_{\ell+1..m}}\ \lambda_{h:\tau\varphi_{1..n-1}\psi_{1..m}\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(q\,y),x{=}(\mathbf{f}_{n'}\,y))))\!:\!\Gamma\ \in\ R_2 \qquad \text{(Aa)}$$

$$\lambda_{g:(\tau\varphi_{1..n-1}\text{-}\mathbf{b}\gamma\psi_{1..\ell}\to\Delta)\to\Gamma}\ \lambda_{q:\gamma\psi_{\ell+1..m}}\ \lambda_{h:\tau\varphi_{1..n-1}\psi_{1..m}\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),(\mathbf{f}_{n'}\,x){=}y)))\!:\!\Gamma\ \in\ R_2 \qquad \text{(Ab)}$$

Cued associations resulting from these inference rules are shown in Figure 10.

Cued-association structures can be derived using these grammatical inference rules in applications L↓, L↑, G↓, and G↑ as defined in Section 3. These derivations start with an empty store of type $\mathbf{T}\to\mathbf{T}$ followed by a sequence of unit tokens with word types, and alternately apply lexical inference rules and grammatical inference rules to consume all the words in the sequence and pro-

duce another store of type $\mathbf{T}\rightarrow\mathbf{T}$ containing a complete corresponding cued-association structure. For example, processing the sentence *Everything works,* using the below lexical inference rules for *everything* and *works*:

$$\lambda_{g:\beta\rightarrow\Gamma}\ \lambda_{w:\textbf{everything}}\ \lambda_{h:\mathbf{N}\rightarrow\beta}\ (g{\circ}h\,(\lambda_x\,\exists_z\,(\mathbf{f_0}\,z){=}\textbf{Every},(\mathbf{f_1}\,z){=}(\mathbf{f_{rin}}\,x),(\mathbf{f_1}\,z){=}x,\ \exists_y\,(\mathbf{f_0}\,y){=}\textbf{BeingAThing},(\mathbf{f_1}\,y){=}(\mathbf{f_{rin}}\,x)))\!:\!\Gamma\ \in\ R_0 \quad (17)$$

$$\lambda_{g:\beta\rightarrow\Gamma}\ \lambda_{w:\textbf{works}}\ \lambda_{h:\textbf{V-aN}\rightarrow\beta}\ (g{\circ}h\,(\lambda_x\,(\mathbf{f_0}{\circ}\mathbf{f_{rin}}\,x){=}\textbf{Working},(\mathbf{f_1}{\circ}\mathbf{f_{rin}}\,x){=}(\mathbf{f_{1'}}\,x)))\!:\!\Gamma\ \in\ R_0 \quad (18)$$

will produce the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\lambda_{q,x_0}\,(q\,x_0)\!:\!\mathbf{T}\rightarrow\mathbf{T}\qquad \textbf{unit}\!:\!\textbf{everything}}{\lambda_{h,x_0}\,h\,(\lambda_{x_1}\,\exists_{z_1}\,(\mathbf{f_0}\,z_1){=}\textbf{Every},(\mathbf{f_1}\,z_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,z_1){=}x_1,\ \exists_{y_1}\,(\mathbf{f_0}\,y_1){=}\textbf{BeingAThing},(\mathbf{f_1}\,y_1){=}(\mathbf{f_{rin}}\,x_1))\!:\!(\mathbf{N}\rightarrow\mathbf{T})\rightarrow\mathbf{T}}\,\mathrm{L}{\downarrow}}{\lambda_{q,h,x_0}\,h\,(\lambda_{x_2}\,\exists_{x_1}\,\exists_{z_1}\,(\mathbf{f_0}\,z_1){=}\textbf{Every},(\mathbf{f_1}\,z_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,z_1){=}x_1,\ \exists_{y_1}\,(\mathbf{f_0}\,y_1){=}\textbf{BeingAThing},(\mathbf{f_1}\,y_1){=}(\mathbf{f_{rin}}\,x_1),\ (q\,x_2),(\mathbf{f_{1'}}\,x_2){=}x_1)\!:\!\textbf{V-aN}\rightarrow(\mathbf{S}\rightarrow\mathbf{T})\rightarrow\mathbf{T}}\,\mathrm{G}{\downarrow_{\mathrm{Aa}}}}{\lambda_{h,x_0}\,h\,(\lambda_{x_2}\,\exists_{x_1}\,\exists_{z_1}\,(\mathbf{f_0}\,z_1){=}\textbf{Every},(\mathbf{f_1}\,z_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,z_1){=}x_1,\ \exists_{y_1}\,(\mathbf{f_0}\,y_1){=}\textbf{BeingAThing},(\mathbf{f_1}\,y_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\textbf{Working},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}x_1,(\mathbf{f_{1'}}\,x_2){=}x_1)\!:\!(\mathbf{S}\rightarrow\mathbf{T})\rightarrow\mathbf{T}\qquad \textbf{unit}\!:\!\textbf{works}}\,\mathrm{L}{\uparrow}}{\lambda_{q,x_0}\,\exists_{x_1,x_2}\,\exists_{z_1}\,(\mathbf{f_0}\,z_1){=}\textbf{Every},(\mathbf{f_1}\,z_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,z_1){=}x_1,\ \exists_{y_1}\,(\mathbf{f_0}\,y_1){=}\textbf{BeingAThing},(\mathbf{f_1}\,y_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\textbf{Working},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}x_1,(\mathbf{f_{1'}}\,x_2){=}x_1,(q\,x_0))\!:\!\mathbf{T}\rightarrow\mathbf{T}}\,\mathrm{G}{\uparrow_{\mathrm{D}}}$$

Cued associations produced by lexical inference rules for *everything* and *works* and as a result of this derivation are shown in Figure 11.

6.3 Auxiliary attachment (binary)

It is also common to attach the signified structure of one sign as the syntactic argument of another structure but make the resulting structure signify the argument. As with argument attachment, if one sign has a type γ and the other has a type consisting of a primitive τ followed by n argument dependencies $\varphi_{1..n}$, the last of which is **-a** or **-b** followed by a γ, then the resulting structure will be of type $\tau\varphi_{1..n-1}$, lacking the final dependency, which is now satisfied. Additionally, any non-local dependencies $\psi_{1..\ell}$ on the first sign and any non-local dependencies $\psi_{\ell+1..m}$ on the second sign will be concatenated and propagated up into the combined sign as $\psi_{1..m}$:

$$\lambda_{g:(\gamma\psi_{1..\ell}\rightarrow\Delta)\rightarrow\Gamma}\ \lambda_{q:\tau\varphi_{1..n-1}\textbf{-a}\gamma\psi_{\ell+1..m}}\ \lambda_{h:\tau\varphi_{1..n-1}\psi_{1..m}\rightarrow\Delta}\ (g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),x{=}(\mathbf{f}_{n'}\,y))))\!:\!\Gamma\ \in\ R_2 \quad \text{(Ua)}$$

$$\lambda_{g:(\tau\varphi_{1..n-1}\textbf{-b}\gamma\psi_{1..\ell}\rightarrow\Delta)\rightarrow\Gamma}\ \lambda_{q:\gamma\psi_{\ell+1..m}}\ \lambda_{h:\tau\varphi_{1..n-1}\psi_{1..m}\rightarrow\Delta}\ (g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(q\,y),(\mathbf{f}_{n'}\,x){=}y)))\!:\!\Gamma\ \in\ R_2 \quad \text{(Ub)}$$

Cued associations resulting from these inference rules are shown in the top row of Figure 10.

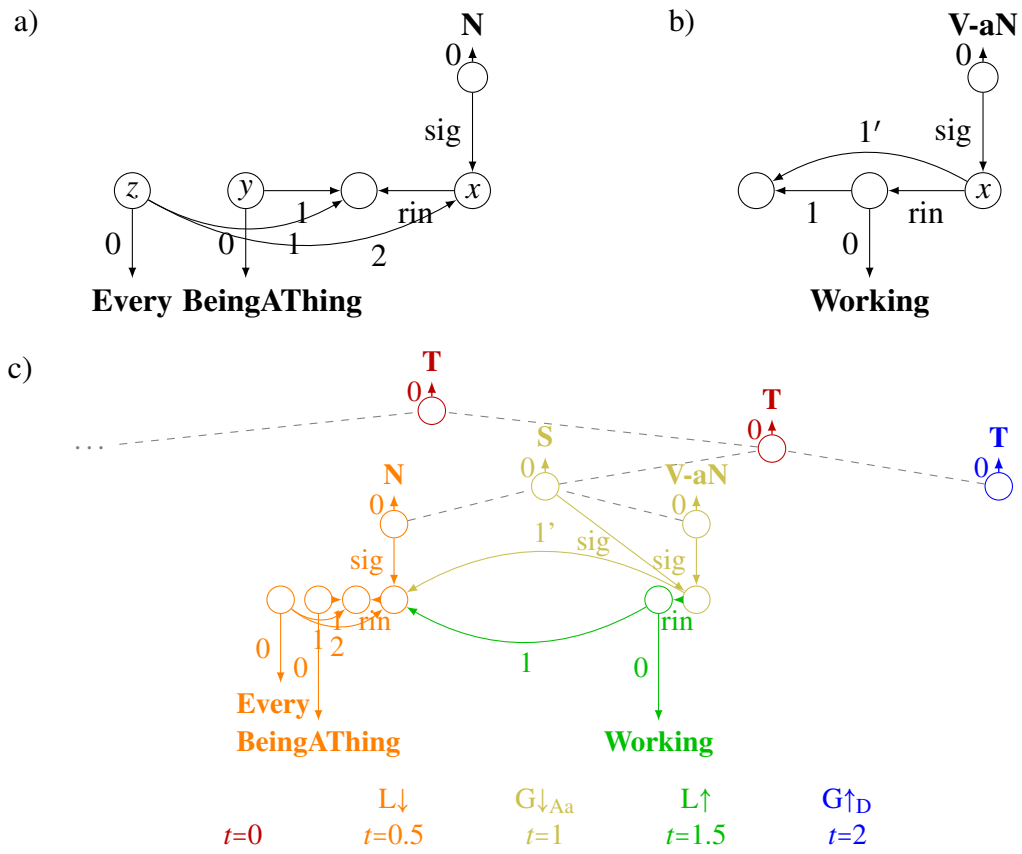


Figure 11: Cued-association structures lexically inferred for words *everything* (a) and *works* (b), and cued associations derived from the sentence *Everything works* (c), with highlights showing associations generated at each derivation step.

Attachment of complements of auxiliaries. Auxiliary attachment rules are used for attachment of complements of auxiliary verbs, which provide the referential state of the resulting phrase. For example, the signified structure of the sign *here* of type **V-aN-b(A-aN)** may be attached to the signified structure of the auxiliary verb *is* of type **A-aN** using an auxiliary attachment rule:

(43) Everything [$_{\textbf{V-aN}}$ [$_{\textbf{V-aN-b(A-aN)}}$ is] [$_{\textbf{A-aN}}$ here]].

A sample derivation of this sentence is shown in Figure 12, and a formal derivation of this sentence is provided in Appendix C.1.

Attachment of complements of markers. Auxiliary attachment rules are also used for attachment of markers like *to* and *that* to complements:

(44) a. Kim wants [$_{\textbf{I-aN}}$ [$_{\textbf{I-aN-b(B-aN)}}$ to] [$_{\textbf{N-aD}}$ help]].
b. Kim thinks [$_{\textbf{C}}$ [$_{\textbf{C-bV}}$ that] [$_{\textbf{V}}$ everything works]].

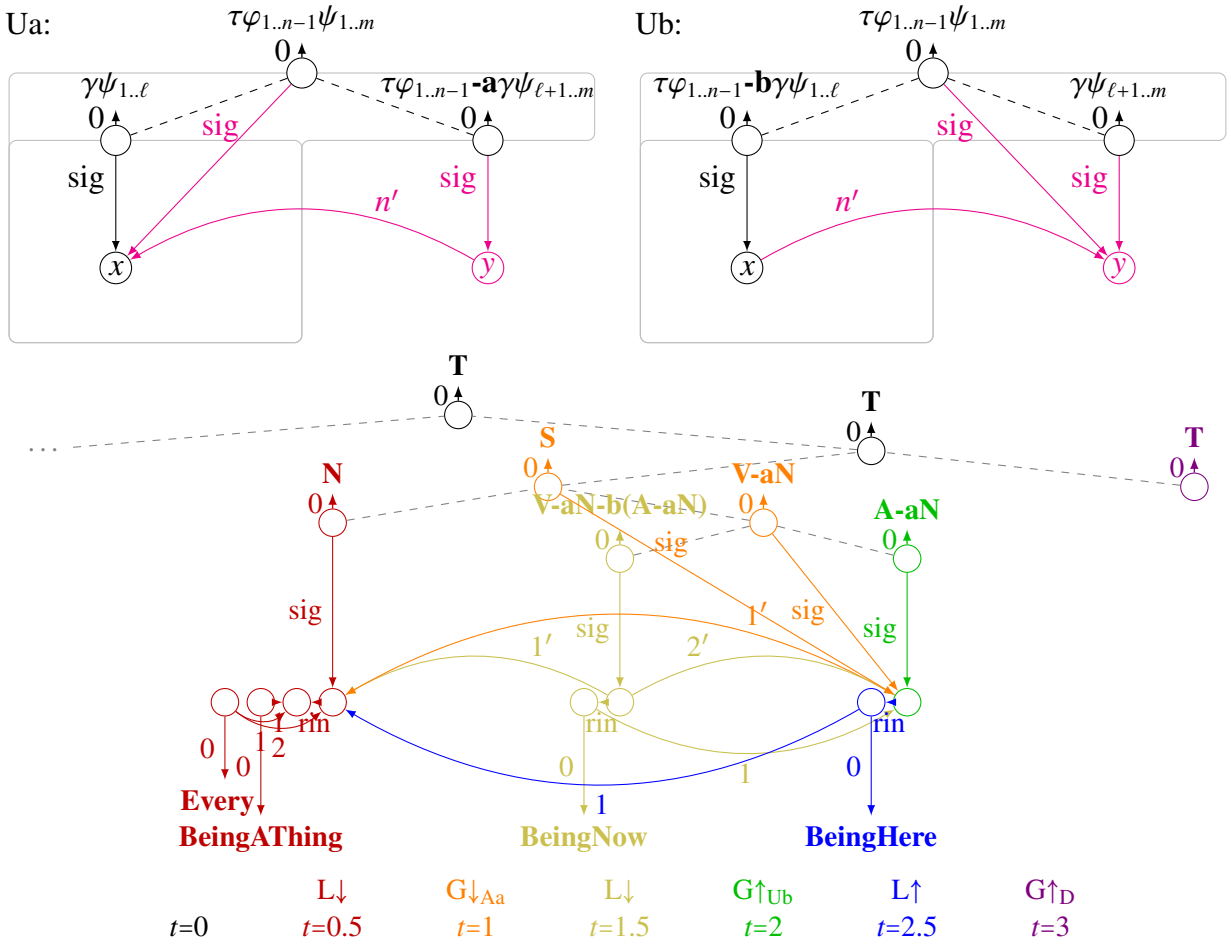


Figure 12: Results of grammatical inference rules for attaching cued-association structures of preceding and succeeding signs as auxiliary structures (top), and cued associations derived from the sentence *Everything is here,* with highlights showing associations generated at each derivation step (bottom).

Attachment of quantifiers and possessive genitives within noun phrases. Auxiliary attachment rules are also used for attachment of quantifiers like *most* and possessive genitives like *Kim's* within noun phrases:

(45) a. [$_{\mathbf{N}}$ [$_{\mathbf{N\text{-}b(N\text{-}aD)}}$ Most] [$_{\mathbf{N\text{-}aD}}$ things]] work.
b. [$_{\mathbf{N}}$ [$_{\mathbf{N\text{-}b(N\text{-}aD)}}$ Kim's] [$_{\mathbf{N\text{-}aD}}$ things]] work.

6.4 Modifier attachment (binary)

It is also possible to attach a sign as a modifier of another sign, such that the constraints of the modifier are applied to the restriction of the modificand rather than the nuclear scope. A modifier sign of type τ-**a**υ, with one argument dependency, can be combined by this rule with a preceding or succeeding modificand sign of any type γ to form a combined sign of type γ that signifies the the

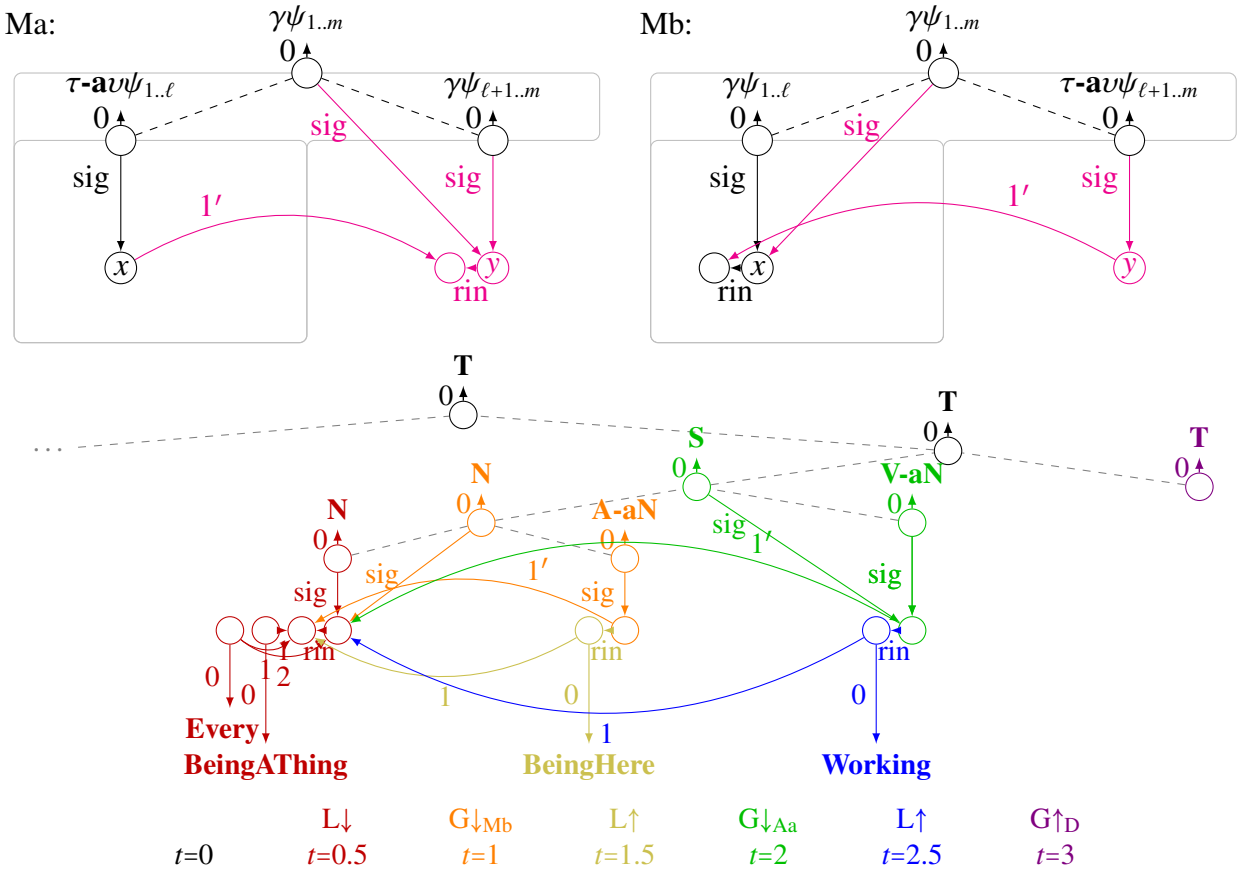


Figure 13: Results of grammatical inference rules for attaching cued-association structures of preceding and succeeding signs as modifiers (top), and cued associations derived from the sentence *Everything here works,* with highlights showing associations generated at each derivation step (bottom).

signified structure of the modificand, with the first syntactic argument of the modifier associated to the restriction of the modificand. As with argument and auxiliary attachment, any non-local dependencies $\psi_{1..\ell}$ on the first sign and any non-local dependencies $\psi_{\ell+1..m}$ on the second sign will be concatenated and propagated up into the combined sign as $\psi_{1..m}$:

$$\lambda_{g:(\tau\text{-}\mathbf{a}\upsilon\psi_{1..\ell}\rightarrow\Delta)\rightarrow\Gamma}\ \lambda_{q:\gamma\psi_{\ell+1..m}}\ \lambda_{h:\gamma\psi_{1..m}\rightarrow\Delta}$$
$$(g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(q\,y),(\mathbf{f}_{\mathbf{1}'}\,x)=(\mathbf{f}_{\mathbf{rin}}\,y))))\!:\!\Gamma\ \in\ R_2 \qquad \text{(Ma)}$$

$$\lambda_{g:(\gamma\psi_{1..\ell}\rightarrow\Delta)\rightarrow\Gamma}\ \lambda_{q:\tau\text{-}\mathbf{a}\upsilon\psi_{\ell+1..m}}\ \lambda_{h:\gamma\psi_{1..m}\rightarrow\Delta}$$
$$(g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),(\mathbf{f}_{\mathbf{rin}}\,x)=(\mathbf{f}_{\mathbf{1}'}\,y))))\!:\!\Gamma\ \in\ R_2 \qquad \text{(Mb)}$$

Cued associations resulting from these inference rules are shown in the top row of Figure 13. Types of modifiers are constrained to have a single dependency to a primitive type, which is not constrained to match the type of the modificand. This unconstrained match allows modifiers to have the same type as certain types of predicates. For example, any predicative phrase **A-aN**:

(46) Everything is [$_{\mathbf{A\text{-}aN}}$ here].

can serve as a post-nominal modifier:

(47) [$_{\mathbf{N}}$ [$_{\mathbf{N}}$ Everything] [$_{\mathbf{A\text{-}aN}}$ here]] works.

A sample derivation of this sentence is shown at the bottom of Figure 13, and a formal derivation of this sentence is provided in Appendix C.2. Note that in the predicative case (using an argument attachment rule), the constraint *here* is applied to the nuclear scope of the quantifier *every*, making the statement false if any thing is not here. In contrast, in the modifier case (using a modifier attachment rule), the constraint *here* is applied to the restrictor, making the statement still possibly true even if a thing is not here.

Attachment of reduced relatives and other post-nominal adjectival modifiers. Any predicative phrase, including adjectival phrases, prepositional phrases, present participle verb phrases, and passive verb phrases, can also be used as a post-nominal modifier. These forms can all be coordinated:

(48) a. [$_{\mathbf{N}}$ [$_{\mathbf{N}}$ Every gear failure] [$_{\mathbf{A\text{-}aN}}$ predicted by Kim and similar to this]] was fixed.
b. [$_{\mathbf{N}}$ [$_{\mathbf{N}}$ Every gear failure] [$_{\mathbf{A\text{-}aN}}$ resulting from wear and outside the pump]] was fixed.

This includes reduced relative clauses, which are analyzed as modifier attachments of present participle and passive phrases.

Attachment of pre-nominal adjectival modifiers. All predicative forms (adjectival phrases, prepositional phrases, present participle verb phrases, and passive verb phrases) may also occur as pre-nominal modifiers, but complements are usually elided (or hyphenated in orthography). These forms can all be coordinated:

(49) a. Every [$_{\mathbf{N\text{-}aD}}$ [$_{\mathbf{A\text{-}aN}}$ predicted and similar] [$_{\mathbf{N\text{-}aD}}$ gear failure]] was fixed.
b. Every [$_{\mathbf{N\text{-}aD}}$ [$_{\mathbf{A\text{-}aN}}$ resulting and outside] [$_{\mathbf{N\text{-}aD}}$ gear failure]] was fixed.

Attachment of adverbial modifiers. Adverbial forms of these predicates can occur and coordinate sentence-finally:

(50) a. A gear [$_{\mathbf{V\text{-}aN}}$ [$_{\mathbf{V\text{-}aN}}$ failed] [$_{\mathbf{R\text{-}aN}}$ similarly to this and predicted by Kim]].
b. A gear [$_{\mathbf{V\text{-}aN}}$ [$_{\mathbf{V\text{-}aN}}$ failed] [$_{\mathbf{R\text{-}aN}}$ resulting from wear and outside the pump]].

and sentence-initially:

(51) a. [$_{\mathbf{S}}$ [$_{\mathbf{R\text{-}aN}}$ Similarly to this and predicted by Kim], [$_{\mathbf{S}}$ a gear failed]].
b. [$_{\mathbf{S}}$ [$_{\mathbf{R\text{-}aN}}$ Resulting from wear and outside the pump], [$_{\mathbf{S}}$ a gear failed]].

and sentence-medially, in the case of adverbs with complements elided:

(52) A gear [$_{\mathbf{V\text{-}aN}}$ [$_{\mathbf{R\text{-}aN}}$ similarly] [$_{\mathbf{V\text{-}aN}}$ failed]].

Adverbial forms may also occur as phrase-initial modifiers to noun phrases:

(53) a. [$_{\mathbf{N}}$ [$_{\mathbf{R\text{-}aN}}$ All and only] [$_{\mathbf{N}}$ the pumps]].

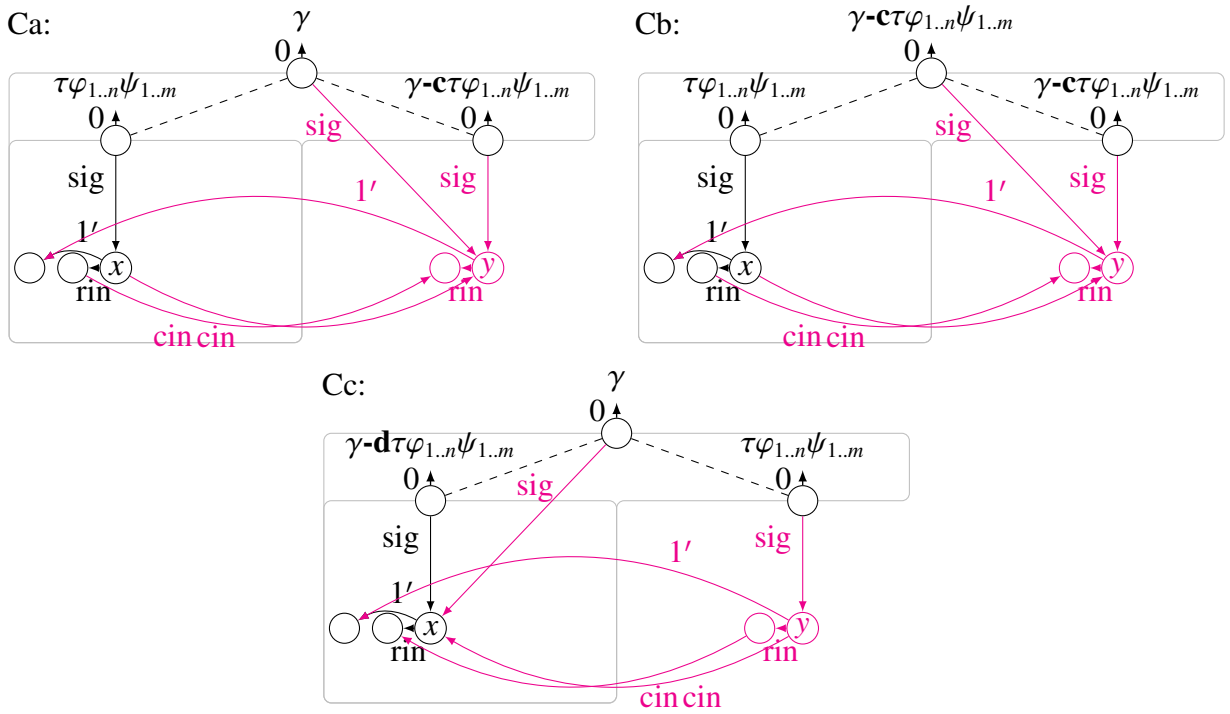


Figure 14: Results of grammatical inference rules for attaching cued-association structures of initial, medial, and final signs as coordinates.

6.5 Coordinate attachment (binary)

Signs can also combine through a process of coordination, in which several coordinates of the same type are conjoined into a larger sign of the same type, and arguments, modifiers, and non-local dependencies above this coordination are shared. Rules for combining initial, medial, and final coordinates are defined below. Each rule for constructing a sign of type $\gamma = \tau\varphi_{1..n}\psi_{1..m}$ from coordinates also of type γ constructs n syntactic argument associations from each coordinate sign coterminal with the same labeled association from the combined sign. Each rule for constructing a sign of type $\tau\varphi_{1..n}\psi_{1..m}$ from coordinates of the same type γ also inherits m non-local dependencies from the store $\lambda_h \ldots$ of type $(\gamma\rightarrow\Delta)\rightarrow\Gamma$. These rules also inherit modifier constraints through a **coordination inheritance** association (notated 'cin'), which behaves like restriction inheritance in quantified structures:

$$\lambda_{g:(\tau\varphi_{1..n}\psi_{1..m}\rightarrow\Delta)\rightarrow\Gamma}\ \lambda_{q:\gamma\text{-}\mathbf{c}\tau\varphi_{1..n}\psi_{1..m}}\ \lambda_{h:\gamma\rightarrow\Delta}\ (g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(q\,y),$$
$$(\mathbf{f}_{\mathbf{1}'}\,x)=(\mathbf{f}_{\mathbf{1}'}\,y),\ldots,(\mathbf{f}_{n'}\,x)=(\mathbf{f}_{n'}\,y),(\mathbf{f}_{\mathbf{cin}}\,x)=y,(\mathbf{f}_{\mathbf{cin}}\circ\mathbf{f}_{\mathbf{rin}}\,x)=(\mathbf{f}_{\mathbf{rin}}\,y))))\,:\Gamma\ \in\ R_2 \qquad \text{(Ca)}$$

$$\lambda_{g:(\tau\varphi_{1..n}\psi_{1..m}\rightarrow\Delta)\rightarrow\Gamma}\ \lambda_{q:\gamma\text{-}\mathbf{c}\tau\varphi_{1..n}\psi_{1..m}}\ \lambda_{h:\gamma\text{-}\mathbf{c}\tau\varphi_{1..n}\psi_{1..m}\rightarrow\Delta}\ (g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(q\,y),$$
$$(\mathbf{f}_{\mathbf{1}'}\,x)=(\mathbf{f}_{\mathbf{1}'}\,y),\ldots,(\mathbf{f}_{n'}\,x)=(\mathbf{f}_{n'}\,y),(\mathbf{f}_{\mathbf{cin}}\,x)=y,(\mathbf{f}_{\mathbf{cin}}\circ\mathbf{f}_{\mathbf{rin}}\,x)=(\mathbf{f}_{\mathbf{rin}}\,y))))\,:\Gamma\ \in\ R_2 \qquad \text{(Cb)}$$

$$\lambda_{g:(\gamma\text{-}\mathbf{d}\tau\varphi_{1..n}\psi_{1..m}\rightarrow\Delta)\rightarrow\Gamma}\ \lambda_{q:\tau\varphi_{1..n}\psi_{1..m}}\ \lambda_{h:\gamma\rightarrow\Delta}\ (g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),$$
$$(\mathbf{f}_{\mathbf{1}'}\,x)=(\mathbf{f}_{\mathbf{1}'}\,y),\ldots,(\mathbf{f}_{n'}\,x)=(\mathbf{f}_{n'}\,y),x=(\mathbf{f}_{\mathbf{cin}}\,y),(\mathbf{f}_{\mathbf{rin}}\,x)=(\mathbf{f}_{\mathbf{cin}}\circ\mathbf{f}_{\mathbf{rin}}\,y))))\,:\Gamma\ \in\ R_2 \qquad \text{(Cc)}$$

Cued associations resulting from these inference rules, for the case where $n = 1$, are shown in Figure 14.

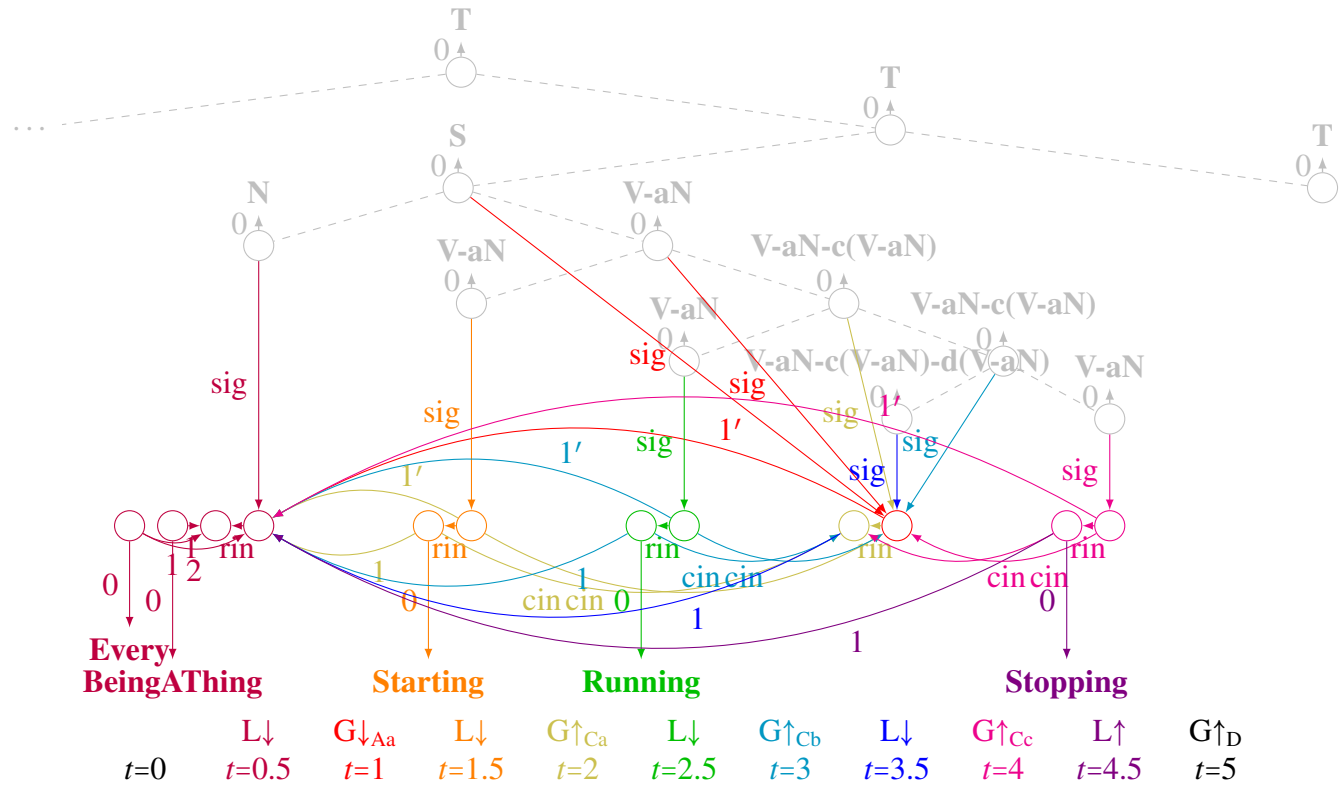


Figure 15: Cued associations derived from the sentence *Everything starts runs and stops*, with highlights showing associations generated at each derivation step.

Coordination of verb phrases and other predicates. Clausal signs, verb phrases and other predicative phrases can all be coordinated. An example derivation of the sentence:

(54) Everything [$_{\textbf{V-aN}}$ [$_{\textbf{V-aN}}$ starts], [$_{\textbf{V-aN-c(V-aN)}}$ [$_{\textbf{V-aN}}$ runs] [$_{\textbf{V-aN-c(V-aN)}}$ and [$_{\textbf{V-aN}}$ stops]]]].

is shown in Figure 15.

Coordination of quantified noun phrases. Quantified noun phrases can also be coordinated:

(55) [$_{\textbf{N}}$ [$_{\textbf{N}}$ Everything] [$_{\textbf{N-cN}}$ and [$_{\textbf{N}}$ everyone]]] works.

This coordination produces the desired meaning, that everything works and everyone works, as shown in Figure 16.

6.6 Argument re-ordering (unary)

In some cases, canonical outer and inner syntactic arguments of signs are re-ordered to signify interrogative mood or to postpose heavily specified signs to the end of a clause. In order to avoid confusing the canonical syntactic argument associations defined by lexical inference rules with the re-ordered syntactic argument associations used in argument attachment, grammatical inference

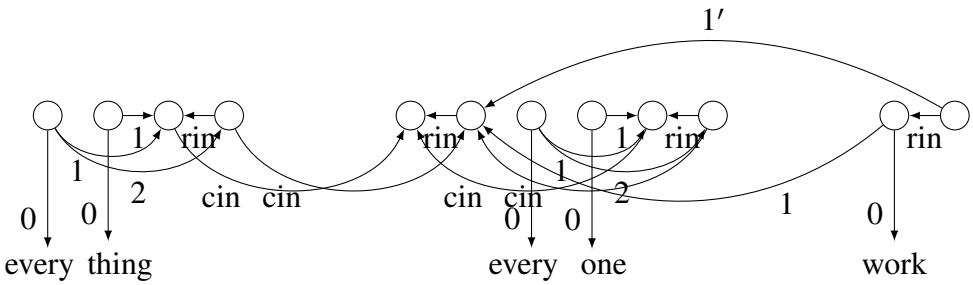


Figure 16: Cued associations derived from the sentence *Everything and everyone works*.

rules for re-ordering these arguments construct new signs with new syntactic argument associations. Signified structures of these new signs are inherited from signified structures of the old signs using an **ordering inheritance** (oin) association. Argument dependencies φ_1 and φ_2 in canonical sign types are then swapped in the re-ordered sign types (in subject-auxiliary inversion), or the first argument of the canonical sign type is moved to the second position of the re-ordered sign type and a new first argument is introduced (in *it* extraposition). Non-local dependencies $\psi_{1..m}$ at the end of the canonical type are propagated up to the re-ordered sign type.

Subject-auxiliary inversion in polar questions and declarative clauses. Subject-auxiliary inversion is used to mark polar questions (which have *yes* or *no* answers):

(56) [**Q-b(A-aN)-bN** [**V-aN-b(A-aN)** are]] you ill?

Subject-auxiliary inversion also occurs in certain declarative contexts, such as following the word *nor*:

(57) You are not well, nor [**Q-b(A-aN)-bN** [**V-aN-b(A-aN)** are]] you ill.

A grammatical inference rule for subject-auxiliary inversion creates a new sign of type $\tau\varphi_2\varphi_1\psi_{1..m}$ which swaps the argument dependencies of the uninverted sign. The signified structure of the new sign also associates syntactic argument associations $1'$ and $2'$ of the uninverted sign as syntactic arguments $2'$ and $1'$, respectively:

$$\lambda_{g:(\gamma\varphi_1\varphi_2\psi_{1..m}\to\Delta)\to\Gamma}\ \lambda_{h:\gamma\varphi_2\varphi_1\psi_{1..m}\to\Delta}$$
$$(g\,(\lambda_p\, h\,(\lambda_y\, \exists_x\,(p\,x), (\mathbf{f}_{\mathbf{1}'}\,x){=}(\mathbf{f}_{\mathbf{2}'}\,y), (\mathbf{f}_{\mathbf{2}'}\,x){=}(\mathbf{f}_{\mathbf{1}'}\,y), (\mathbf{f}_{\mathbf{oin}}\,y){=}x)))):\Gamma\ \in\ R_1 \qquad \text{(Oa)}$$

Cued associations resulting from this inference rule are shown in Figure 17a.

***It* extraposition.** *It* extraposition uses an expletive subject *it* and adds a complement which is interpreted as the subject:

(58) It [**V-aNe-bC** [**V-aN** bothers me]] that Kim left.

A grammatical inference rule for *it* extraposition creates a new sign of type τ**-aNe**$\varphi_1\psi_{1..m}$ which moves the subject dependency of the uninverted sign to a later position in the clause. The signified

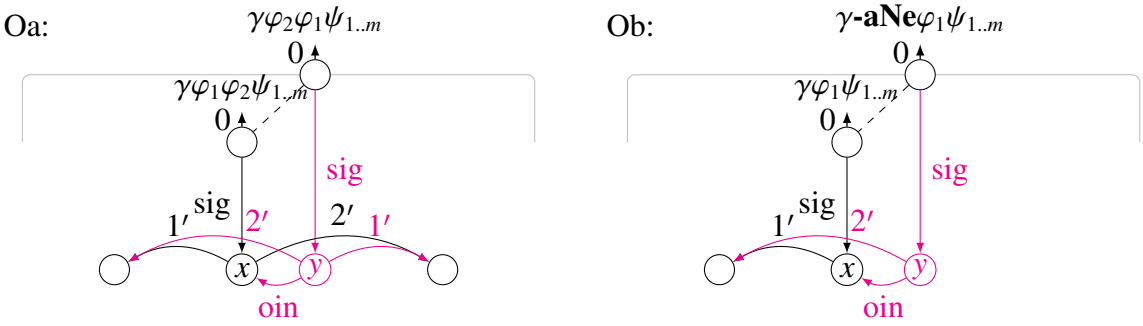


Figure 17: Results of grammatical inference rules for subject-auxiliary inversion (a) and *it* extraposition (b).

structure of the new sign also associates syntactic argument association $1'$ of the uninverted sign as syntactic argument $2'$:

$$\lambda_{g:(\gamma\varphi_1\psi_{1..m}\rightarrow\Delta)\rightarrow\Gamma}\ \lambda_{h:\gamma\text{-aNe}\varphi_1\psi_{1..m}\rightarrow\Delta}$$
$$(g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(\mathbf{f}_{1'}\,x){=}(\mathbf{f}_{2'}\,y),(\mathbf{f}_{\mathbf{oin}}\,y){=}x)))\,{:}\,\Gamma\ \in\ R_1 \qquad \text{(Ob)}$$

Cued associations resulting from this inference rule, are shown in Figure 17b.

6.7 Extraction and non-local dependency elimination (unary)

Non-local dependencies allow signs that are not adjacent in a derivation structure to be syntactically associated. In a sentence processing model, these non-local dependencies can be defined by extending the operations of a pushdown store to allow access to items that are not at the head of the store. These random-access items are signs typed with non-local dependencies ψ that may occur between the top and bottom signs of a derivation fragment, connected by 'A' associations, or between the bottom and top signs of successive derivation fragments, connected by 'B' associations. Dependencies $\psi_{1..m}$ to required non-local signs are appended onto sign types after primitive types τ and ordinary syntactic argument dependencies $\varphi_{1..n}$, and are needed to accurately distinguish sign types for purposes of substitution and coordination.

Non-local dependencies may be lexically specified, as is the case with relative and interrogative pronouns, or may be converted from syntactic arguments or introduced as modifiers using unary grammatical inference rules. These rules may add new non-local signs to the store or access existing non-local signs from the store. In the former case, non-local signs are added to the store through a 'B' association. In the latter case, these rules may allow an arbitrary number of 'A' or 'B' associations to be traversed in order to access non-local signs, essentially cueing these signs by content.

Non-local dependencies are propagated up to types of superordinate signs in argument attachment, auxiliary attachment, modifier attachment, and other rules. While non-local dependencies must be copied to sign types that result from these rules, the non-local signs themselves and the cued-association structures signified by these signs are propagated simply by remaining on the store while these rules apply.

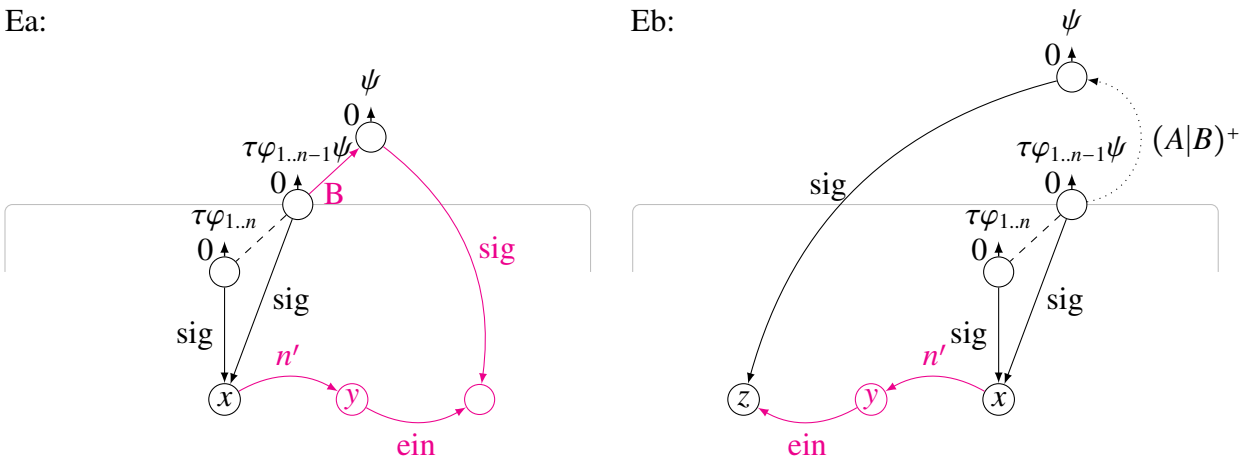


Figure 18: Results of grammatical inference rules for converting syntactic argument dependencies to non-local dependencies.

When signs with non-local dependencies are coordinated, the signified structure of a non-local sign may become shared. If this shared structure is an elementary predication with a syntactic associations to structures signified by both of the coordinates (as is the case with modifier extraction), then it will not be possible to distinguish these multiple destinations from ambiguity, which is undesirable. In order to prevent this, coordinates instead inherit from shared structures using **extraction inheritance** ('ein') associations. These new inheritances behave in much the same way as restriction inheritances ('rin') and coordination inheritances ('cin'), allowing constraints from destination referential states to be inherited to source referential states.

Extraction of arguments. Arguments may be extracted by converting the last syntactic argument dependency of a sign of type $\tau\varphi_{1..n}$ with n syntactic argument dependencies into a non-local dependency ψ. Grammatical inference rules for argument extraction construct an extraction inheritance ('ein') association from the n^{th} syntactic argument y of the signified structure of this sign to the signified structure of a new or existing non-local sign.

If a new non-local sign is introduced, it is connected to its host sign by a 'B'-labeled cued association:

$$\lambda_{g:(\tau\varphi_{1..n}\to\beta)\to\Gamma}\ \lambda_{h:\tau\varphi_{1..n-1}\psi\to\beta}\ \lambda_{q':\psi}\ (g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q'\,(\mathbf{f}_{\mathbf{ein}}\,y)),(\mathbf{f}_{n'}\,x){=}y)))\!:\!\Gamma\ \in\ R_1 \qquad \text{(Ea)}$$

Cued associations resulting from this inference rule are shown on the left in Figure 18. New arguments are extracted in cases where a heavy argument is postposed:

(59) Kim has not [$_{\textbf{L-aN-hN}}$ [$_{\textbf{L-aN-bN}}$ told Pat] _] yet any secrets of the Ada programming language.

If an existing non-local sign is accessed, it may be retrieved via an unbounded number of 'A' and 'B' associations on the store:

$$\lambda_{g:(\tau\varphi_{1..n}\to\Delta)\to\ldots\to\psi\to\Gamma}\ \lambda_{h:\tau\varphi_{1..n-1}\psi\to\Delta}\ \ldots\ \lambda_{q':\psi}$$
$$\exists_y\,(g\,(\lambda_p\,h\,(\lambda_x\,(p\,x),(\mathbf{f}_{n'}\,x){=}y))\ \ldots\ (\lambda_z\,(q'\,z),(\mathbf{f}_{\mathbf{ein}}\,y){=}z))\!:\!\Gamma\ \in\ R_1 \qquad \text{(Eb)}$$

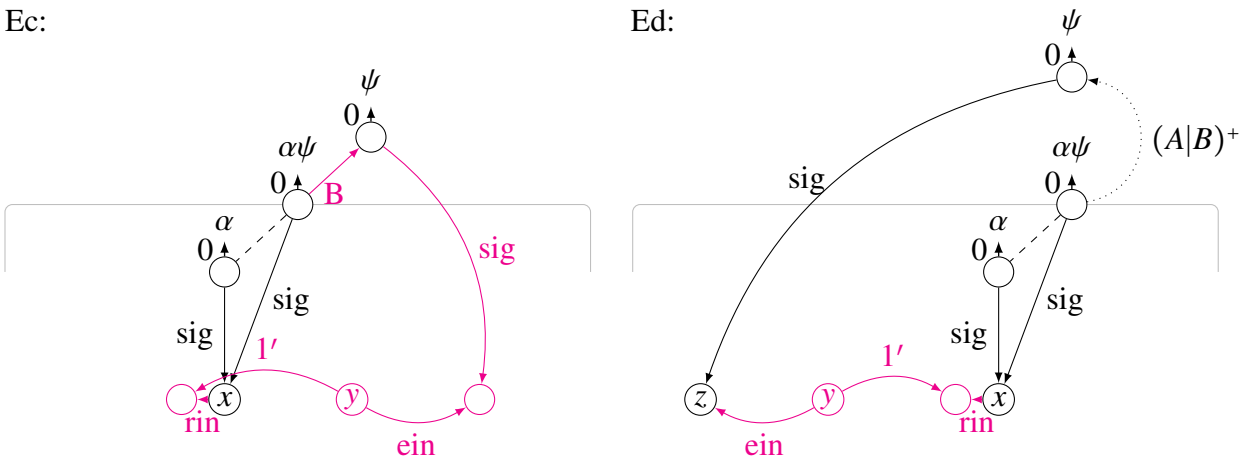


Figure 19: Results of grammatical inference rules for introducing modifiers as non-local dependencies.

Cued associations resulting from this inference rule are shown on the right in Figure 18. Existing arguments are extracted in cases of filler-gap constructions within questions or relative clauses, and in cases of passive constructions:

(60) a. That's the book that Kim [$_{\textbf{V-aN-gN}}$ [$_{\textbf{V-aN-bN}}$ found] _].
b. Ada is considered [$_{\textbf{A-vN}}$ _ [$_{\textbf{A-aN}}$ object-oriented]].

Extraction of modifiers. Modifiers may be extracted from any sign by adding a non-local dependency for a modifier. Grammatical inference rules for modifier extraction construct a new referential state for an elementary predication y and then construct a syntactic argument association from this state to the signified structure of the host sign and construct an extraction inheritance ('ein') association from this new predication y to the signified structure of a new or existing non-local sign.

If a new non-local sign is introduced, it is connected to its host sign by a 'B'-labeled cued association:

$$\lambda_{g:(\alpha\to\beta)\to\Gamma}\ \lambda_{h:\alpha\psi\to\beta}\ \lambda_{q':\psi}\ (g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q'\,(\mathbf{f}_{\textbf{ein}}\,y)),(\mathbf{f}_{\textbf{rin}}\,x){=}(\mathbf{f}_{\mathbf{1}'}\,y))))\!:\!\Gamma\ \in\ R_1 \qquad \text{(Ec)}$$

Cued associations resulting from this inference rule are shown on the left in Figure 19. New arguments are extracted in cases where a heavy modifier is postposed:

(61) Kim read [$_{\textbf{N-h(R-aN)}}$ [$_{\textbf{N}}$ a book _]] today about Ada.

If an existing non-local sign is accessed, it may be retrieved via an unbounded number of 'A' and 'B' associations on the store:

$$\lambda_{g:(\tau\varphi_{1..n}\to\Delta)\to\ldots\to\psi\to\Gamma}\ \lambda_{h:\tau\varphi_{1..n-1}\psi\to\Delta}\ \ldots\ \lambda_{q':\psi}$$
$$\exists_y\,(g\,(\lambda_p\,h\,(\lambda_x\,(p\,x),(\mathbf{r}_{\textbf{rin}}\,x){=}(\mathbf{f}_{1'}\,y)))\ \ldots\ (\lambda_z\,(q'\,z),(\mathbf{f}_{\textbf{ein}}\,y){=}z))\!:\!\Gamma\ \in\ R_1 \qquad \text{(Ed)}$$

Cued associations resulting from this inference rule are shown on the right in Figure 19. Existing modifier are extracted in cases of filler-gap constructions within questions or relative clauses:

(62) Where did Kim think Pat [$_{\textbf{V-aN-g(R-aN)}}$ [$_{\textbf{V-aN}}$ slept] _]?

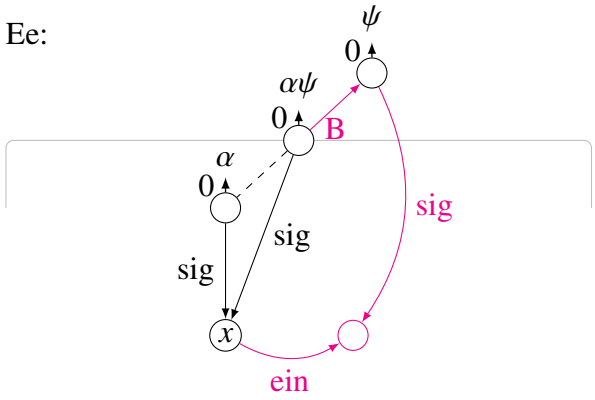


Figure 20: Results of grammatical inference rules for introducing relative clauses as non-local dependencies.

Extraction of relative clauses. As with modifiers, relative clauses may be extracted from any sign by adding a non-local dependency for a relative clause. Grammatical inference rules for relative clauses extraction construct an extraction inheritance ('ein') association from the signified structure of the host sign to the signified structure of a new non-local sign. Non-local relative clauses signs are connected to their host signs by a 'B'-labeled cued association:

$$\lambda_{g:(\alpha\to\beta)\to\Gamma}\ \lambda_{h:\alpha\psi\to\beta}\ \lambda_{q':\psi}\ (g\,(\lambda_p\,h\,(\lambda_x\,(p\,x),(q'\,(\mathbf{f}_{\mathbf{ein}}\,x)))))\!:\!\Gamma\ \in\ R_1 \qquad \text{(Ee)}$$

Cued associations resulting from this inference rule are shown in Figure 20. New non-local relative clauses are extracted in cases where a heavy relative clause is postposed:

(63) Kim read $[_{\textbf{N-h(C-rN)}}\ [_{\textbf{N}}$ a book _$]]$ today that Pat wrote.

Non-local dependency elimination. Non-local dependencies can be used more than once, so they should not be removed when they are accessed. It is therefore necessary to define a separate unary grammatical inference rule to remove non-local dependencies. This rule is learned so as to apply only to non-local signs that no longer occur as non-local dependencies in the store (i.e. if $\psi \notin \alpha,\beta,\alpha',\beta',\ldots$):

$$\lambda_{g:(\alpha\to\beta)\to(\alpha'\to\beta')\to\ldots\to\psi\to\Gamma}\ \lambda_{h:\alpha\to\beta,\ h':\alpha'\to\beta',\ldots}\ (g\,h\,h'\ldots\,(\lambda_z\,\mathbf{true}))\!:\!\Gamma\ \in\ R_1 \qquad \text{(N)}$$

6.8 Gap-filler attachment (binary)

Gap fillers are signs that provide semantic constraints for extracted argument and modifier signs. They are preposed to clause initial position and usually have discourse prominence.

Grammatical inference rules for gap-filler attachment combine filler signs of type δ with filler-dependent signs of type γ**-g**δ to form signs of type γ. Non-local dependencies are propagated from the type of the filler to the type of the resulting sign, and the signified structure of the resulting sign is the signified structure of the filler-dependent sign. This rule also adds a non-local gap-filler sign of type **-g**δ to the pushdown store through an 'A' association, so that it is available for extraction in

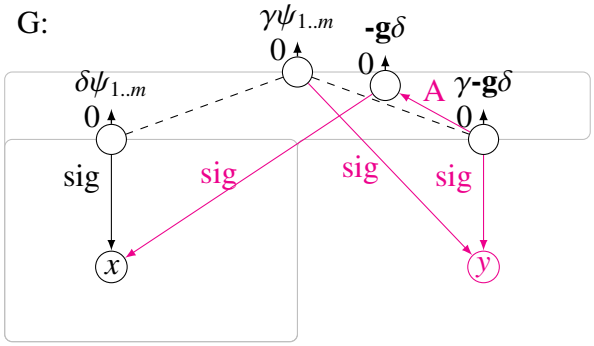


Figure 21: Results of grammatical inference rules for attaching gap fillers.

subordinate signs:

$$\lambda_{g:(\delta\psi_{1..m}\to\Delta)\to\Gamma}\ \lambda_{q:\gamma\text{-}\mathbf{g}\delta}\ \lambda_{q':\text{-}\mathbf{g}\delta}\ \lambda_{h:\gamma\psi_{1..m}\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(q\,y),(q'\,x))))\!:\!\Gamma\ \in\ R_2 \qquad \text{(G)}$$

Cued associations resulting from this inference rule are shown in Figure 21.

Topicalization. Arguments and modifiers can be preposed to clause initial position to make them more prominent in a discourse:

(64) a. [$_{\mathbf{V}}$ [$_{\mathbf{N}}$ Everything here] [$_{\mathbf{V\text{-}gN}}$ someone likes _]].
b. [$_{\mathbf{V}}$ [$_{\mathbf{R\text{-}aN}}$ For that] [$_{\mathbf{V\text{-}g(R\text{-}aN)}}$ they think Kim received a warning _]].

A sample derivation of Sentence 64a is shown in Figure 22, and a formal derivation of this sentence is provided in Appendix C.3.

Attachment of interrogative phrases. Gap-filler attachment is used to attach interrogative phrases within content questions. The requirement for an interrogative pronoun antecedent is propagated up to the resulting sign:

(65) a. [$_{\mathbf{Q\text{-}iN}}$ [$_{\mathbf{N\text{-}iN}}$ What] [$_{\mathbf{Q\text{-}gN}}$ did you find _]]?
b. [$_{\mathbf{Q\text{-}iN}}$ [$_{\mathbf{R\text{-}aN\text{-}iN}}$ Where] [$_{\mathbf{Q\text{-}g(R\text{-}aN)}}$ did you find apples _]]?

Attachment of relative phrases. Gap-filler attachment is also used to attach relative phrases within relative clauses: The requirement for a relative pronoun antecedent is propagated up to the resulting sign:

(66) a. It rained, [$_{\mathbf{V\text{-}rN}}$ [$_{\mathbf{N\text{-}rN}}$ which] [$_{\mathbf{V\text{-}gN}}$ nobody liked _]].
b. They left early, [$_{\mathbf{V\text{-}rN}}$ [$_{\mathbf{R\text{-}aN\text{-}rN}}$ for which] [$_{\mathbf{V\text{-}g(R\text{-}aN)}}$ they received a warning _]].

6.9 Interrogative clause attachment (binary)

Some signs take syntactic arguments that contain non-local dependencies. In some cases the syntactic argument of the signified referential state of the argument-taking sign is associated with the

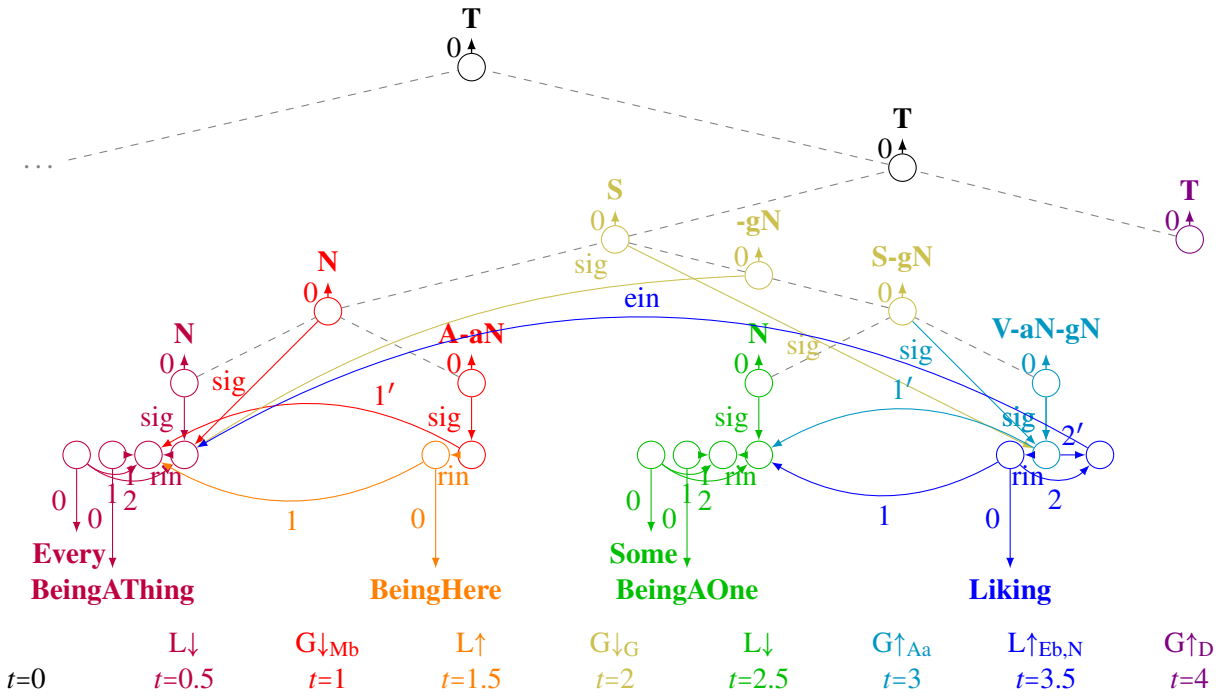


Figure 22: Cued associations derived from the sentence *Everything here someone likes*.

signified referential state of the non-local sign:

$$\lambda_{g:(\tau\varphi_{1..n-1}\text{-}\mathbf{b}(\gamma\psi)\psi_{1..m}\to\Delta)\to\Gamma}\ \lambda_{q:\gamma\psi}\ \lambda_{q':\psi}\ \lambda_{h:\tau\varphi_{1..n-1}\psi_{1..m}\to\Delta}$$
$$(g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),(q'\,(\mathbf{f}_{n'}\,x)))):\Gamma\ \in\ R_2 \qquad \text{(Ia)}$$

Cued associations resulting from this inference rule are shown in Figure 23a. In some cases the syntactic argument of the signified referential state of the argument-taking sign is associated with the signified referential state of the argument sign and the non-local sign signifies the referential state signified by the subject:

$$\lambda_{g:(\tau\varphi_{1..n-1}\text{-}\mathbf{b}(\gamma\psi)\psi_{1..m}\to\Delta)\to\Gamma}\ \lambda_{q:\gamma\psi}\ \lambda_{q':\psi}\ \lambda_{h:\tau\varphi_{1..n-1}\psi_{1..m}\to\Delta}$$
$$(g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),(q'\,(\mathbf{f}_{1'}\,x)),(\mathbf{f}_{n'}\,x){=}y))):\Gamma\ \in\ R_2 \qquad \text{(Ib)}$$

Cued associations resulting from this inference rule are shown in Figure 23b. This kind of attachment is used to connect subordinate interrogative clauses, free relatives and tough constructions. If one sign has a type γ and the other has a type consisting of a primitive τ followed by n argument dependencies $\varphi_{1..n}$, the last of which is **-a** or **-b** followed by a γ, then the signified structure of the sign of type γ may be attached to the signified structure of the other sign by a syntactic argument association with label n'. The resulting structure will be of type $\tau\varphi_{1..n-1}$, lacking the final dependency, which is now satisfied. Additionally, any non-local dependencies $\psi_{1..m}$ on the first sign are propagated up into the combined sign. As with argument attachment, the signified referential state of the resulting sign is the signified state of the non-argument sign.

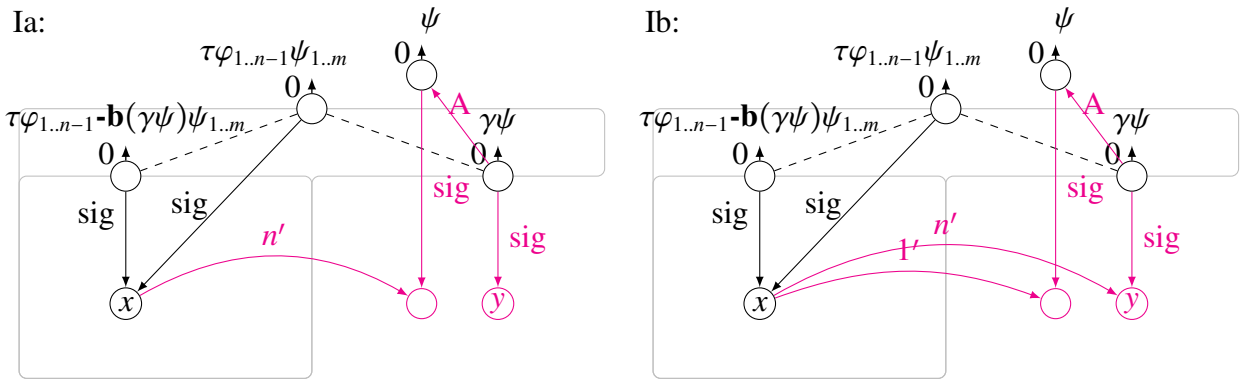


Figure 23: Results of grammatical inference rules for attaching arguments containing non-local signs.

Attachment of subordinate interrogative clauses. Interrogative attachment rule Ia is used to attach signified referential states of antecedents of interrogative pronouns in subordinate interrogative clauses (of type **V-iN**) as syntactic arguments of verbs like *ask* and *wonder*:

(67) Kim [$_{\textbf{V-aN}}$ [$_{\textbf{V-aN-b(V-iN)}}$ wondered] [$_{\textbf{V-iN}}$ which pump Pat checked]].

Note that subordinate interrogative clauses are not subject-auxiliary inverted.

Attachment of clauses containing gap-fillers in free relative clauses. Interrogative attachment rule Ia is also used to attach clauses containing unsatisfied gap-filler dependencies to interrogative pronouns in free relative clauses:

(68) [$_{\textbf{N}}$ [$_{\textbf{N-b(V-gN)}}$ What] [$_{\textbf{V-gN}}$ Kim checked]] was the pump.

The use of an argument containing an unsatisfied filler-gap dependency is appropriate, since arbitrary interrogative phrases are not attested in free relative clauses:

(69) * [$_{\textbf{N}}$ [$_{\textbf{N-iN}}$ Whose pump] [$_{\textbf{V-gN}}$ Kim checked]] was Pat.

Attachment of arguments in tough constructions. Interrogative attachment rule Ib is used to attach infinitive verb phrases containing unsatisfied gap-filler dependencies (of type **I-aN-gN**) as syntactic arguments of adjectives like *easy* and *tough*:

(70) Those pumps are [$_{\textbf{A-aN}}$ [$_{\textbf{A-aN-b(I-aN-gN)}}$ easy] [$_{\textbf{I-aN-gN}}$ to check]].

In these constructions, it is the signified referential state of the argument infinitive verb phrase itself, rather than the signified state of the non-local sign, that is used as the syntactic argument of the adjective. The signified referential state of the non-local dependency is instead associated with the subject of the adjective. In the lexical semantics of the adjective, the referential state of the infinitive verb phrase is then the first and only semantic participant of the elementary predication signified by the adjective.

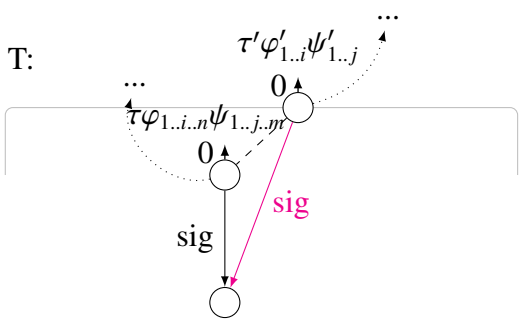


Figure 24: Results of grammatical inference rules for changing types.

6.10 Type change (unary)

In many cases, the type of sign can be coerced into a different type, for example to match an argument of a superordinate sign, without changing the meaning or argument structure of the original sign. This change may preserve the argument dependencies $\varphi_{1..n}$ and non-local dependencies $\psi_{1..m}$ of the original sign type without re-ordering them, or it may elide some argument dependencies $\varphi_{i+1..n}$ or non-local dependencies $\psi_{j+1..m}$ (but again, without re-ordering them):

$$\lambda_{g:(\tau\varphi_{1..i..n}\psi_{1..j..m}\to\Delta)\to\ldots\to(\ldots\to\psi_m\to\ldots)\to\ldots\to(\ldots\to\psi_1\to\ldots)\to\Gamma}$$
$$g:(\tau'\varphi'_{1..i}\psi'_{1..j}\to\Delta)\to\ldots\to(\ldots\to\psi'_j\to\ldots)\to\ldots\to(\ldots\to\psi'_1\to\ldots)\to\Gamma\ \in\ R_1 \qquad \text{(T)}$$

Cued associations resulting from this inference rule are shown in Figure 24.

Converting bare relatives, *that* relatives and *wh* relatives into relative clauses. This type-changing rule is used to coerce bare relatives (**V-gN**), *that* relatives (**C-gN**), and *wh* relatives (**V-rN**) into relative clause type (**C-rN**):

(71) a. Something [$_{\textbf{C-rN}}$ [$_{\textbf{V-gN}}$ everyone likes]] is here.
b. Something [$_{\textbf{C-rN}}$ [$_{\textbf{C-gN}}$ that everyone likes]] is here.
c. Something [$_{\textbf{C-rN}}$ [$_{\textbf{V-rN}}$ which everyone likes]] is here.

English seems to allow restrictive relative clauses to be conjoined, which suggests that they belong to the same type:

(72) Every plan [$_{\textbf{C-rN}}$ [$_{\textbf{C-rN}}$ that the board approves] and [$_{\textbf{C-rN}}$ which is on the list]] is usable.

However, restrictive relative clauses cannot be conjoined with modifiers, suggesting a different type:

(73) * Every plan [$_{?}$ [$_{\textbf{C-rN}}$ that the board approves] and [$_{\textbf{A-aN}}$ on the list]] is usable.

Converting matrix clause types to sentence types. This rule also allows declaratives (**V**), closed (polar) interrogatives (**Q**), open (content) interrogatives (**Q-iN**) and imperatives (**B-aN**) to be converted into sentence types (**S**):

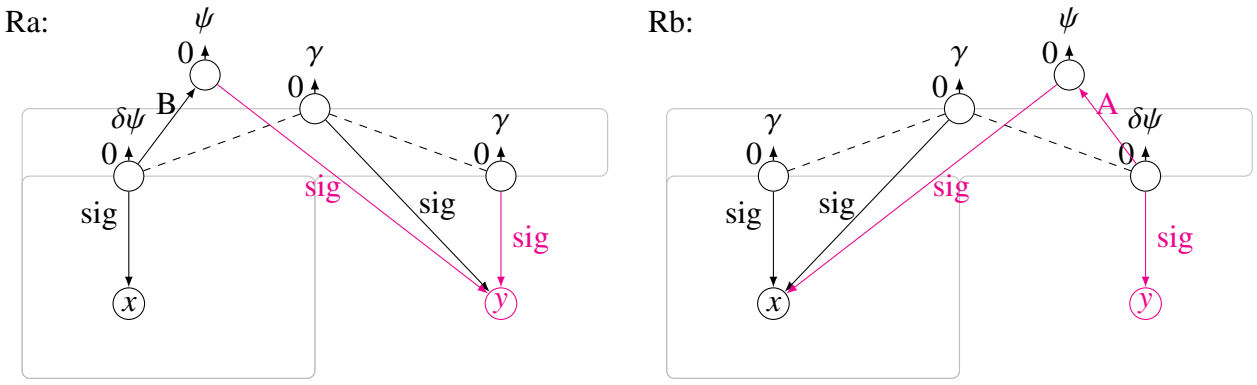


Figure 25: Results of grammatical inference rules for relative clause attachment.

(74) a. [S [V We are here]].
b. [S [Q Are we here]]?
c. [S [Q-iN Where are we]]?
d. [S [B-aN Come here]]!

Note that in the case of open (content) interrogatives (**Q-iN**) the interrogative pronoun dependency is elided, and in the case of imperative clauses (**B-aN**) the subject argument is elided. The signified referential states associated with these dependencies are specified from the discourse to be the queried content or the addressee of the utterance.

Elision of arguments. This type-chainging rule also allows transitive verbs like *called* of type **V-aN-bN** or plural common nouns like *birds* of type **N-aD** to elide their direct objects or their determiners, becoming **V-aN** or **N**:

(75) a. Kim owns [N [N-aD birds]].
b. Kim [V-aN [V-aN-bN called]].

6.11 Relative clause attachment (binary)

A sign may be modified by a relative clause, which (in the case of *wh* relatives) contains a relative pronoun that uses the modificand of the relative clause as an antecedent, or (in *that* relatives and bare relatives) contains a non-local dependency signifying the referential state signified by the modificand. Preceding relative clauses with relative clause or filler-gap dependency ψ introduce a non-local sign ψ onto the pushdown store. Succeeding relative clauses with relative clause or filler-gap dependency ψ match a non-local sign ψ onto the pushdown store. The type and signified state of the modificand of a relative clause is always propagated up to the resulting sign:

$$\lambda_{g:(\delta\psi\to\psi\to\Delta)\to\Gamma}\ \lambda_{q:\gamma}\ \lambda_{h:\gamma\to\Delta}\ (g\,(\lambda_p\,\lambda_{q'}\,h\,(\lambda_y\,\exists_x\,(p\,x),(q\,y),(q'\,y))))\!:\!\Gamma\ \in\ R_2 \qquad \text{(Ra)}$$

$$\lambda_{g:(\gamma\to\Delta)\to\Gamma}\ \lambda_{q:\delta\psi}\ \lambda_{q':\psi}\ \lambda_{h:\gamma\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),(q'\,x))))\!:\!\Gamma\ \in\ R_2 \qquad \text{(Rb)}$$

Cued associations resulting from this inference rule are shown in Figure 25.

Attachment of relative clauses. Relative clauses that modify noun phrases have type **C-rN**, which are type-changed from bare (**V-gN**), *that* (**C-gN**) and *wh* (**V-rN**) relatives.

(76) [$_{\mathbf{N}}$ [$_{\mathbf{N}}$ Something] [$_{\mathbf{C\text{-}rN}}$ everyone likes]] is here.

Supplementary relative clauses that modify other clauses are always *wh* relatives, of type **V-rN**:

(77) The pumps [$_{\mathbf{V\text{-}aN}}$ [$_{\mathbf{V\text{-}aN}}$ were damaged] [$_{\mathbf{V\text{-}rN}}$ which was unfortunate]].

Integrated (restrictive) and supplementary (non-restrictive) relative clauses can be distinguished lexically:

$$\lambda_{g:\Delta\to(\alpha'\to\Delta')\to(\alpha''\to\Delta'')\to\ldots\to\mathbf{\text{-}rN}\to\Gamma}\ \lambda_{w:\mathbf{which}}\ \lambda_{h:\mathbf{N\text{-}rN}\to\Delta,\ h':\alpha'\to\Delta',\ h'':\alpha''\to\Delta'',\ \ldots}$$
$$\exists_y\,(g\,(h\,(\lambda_x\,(\mathbf{f_{ein}}\circ\mathbf{f_{rin}}\,x){=}y))\,h'\,h''\,\ldots\,(\lambda_z\,(\mathbf{f_{rin}}\,z){=}y)):\Gamma\ \in\ R_0 \qquad (19)$$

$$\lambda_{g:\Delta\to(\alpha'\to\Delta')\to(\alpha''\to\Delta'')\to\ldots\to\mathbf{\text{-}rN}\to\Gamma}\ \lambda_{w:\mathbf{which}}\ \lambda_{h:\mathbf{N\text{-}rN}\to\Delta,\ h':\alpha'\to\Delta',\ h'':\alpha''\to\Delta'',\ \ldots}$$
$$\exists_y\,(g\,(h\,(\lambda_x\,(\mathbf{f_{ein}}\circ\mathbf{f_{rin}}\,x){=}y))\,h'\,h''\,\ldots\,(\lambda_z\,z{=}y)):\Gamma\ \in\ R_0 \qquad (20)$$

or using a type-changing rule:

$$\lambda_{g:(\mathbf{V\text{-}gN}\to\Delta)\to(\alpha'\to\Delta')\to(\alpha''\to\Delta'')\to\ldots\to\mathbf{\text{-}gN}\to\Gamma}\ \lambda_{h:\mathbf{C\text{-}rN}\to\Delta,\ h':\alpha'\to\Delta',\ h'':\alpha''\to\Delta'',\ \ldots,\ q':\mathbf{\text{-}rN}}$$
$$(g\,(\lambda_p\,(h\,(\lambda_x\,(p\,x))))\,h'\,h''\,\ldots\,(\lambda_z\,\exists_y(q'\,y),(\mathbf{f_{rin}}\,y){=}z)):\Gamma\ \in\ R_1 \qquad (21)$$

This analysis generalizes several forms of pied piping of relative pronoun dependencies through argument and modifier compositions:

(78) That's the person [$_{\mathbf{C\text{-}rN}}$ [$_{\mathbf{N\text{-}rN}}$ a story [$_{\mathbf{A\text{-}aN\text{-}rN}}$ about [$_{\mathbf{N\text{-}rN}}$ [$_{\mathbf{D\text{-}rN}}$ whose] life]]] was in the paper].

including those introducing quantifiers over the referent of a relative pronoun:

(79) There were ten pumps [$_{\mathbf{C\text{-}rN}}$ [$_{\mathbf{N\text{-}rN}}$ half of which] were damaged].

Attachment of parenthetical clauses containing gaps. This also generalizes gapped parenthetical clause attachment to the left and right of clauses:

(80) a. [$_{\mathbf{S}}$ Tomorrow [$_{\mathbf{S\text{-}gC}}$ they thought _] would be different].
b. [$_{\mathbf{S}}$ Tomorrow would be different [$_{\mathbf{S\text{-}gC}}$ they thought _]].

6.12 Heavy-shift or postposing attachment (binary)

It is possible for a heavy sign to be shifted or postposed to a position later the same clause or to a position later in a superordinate clause.

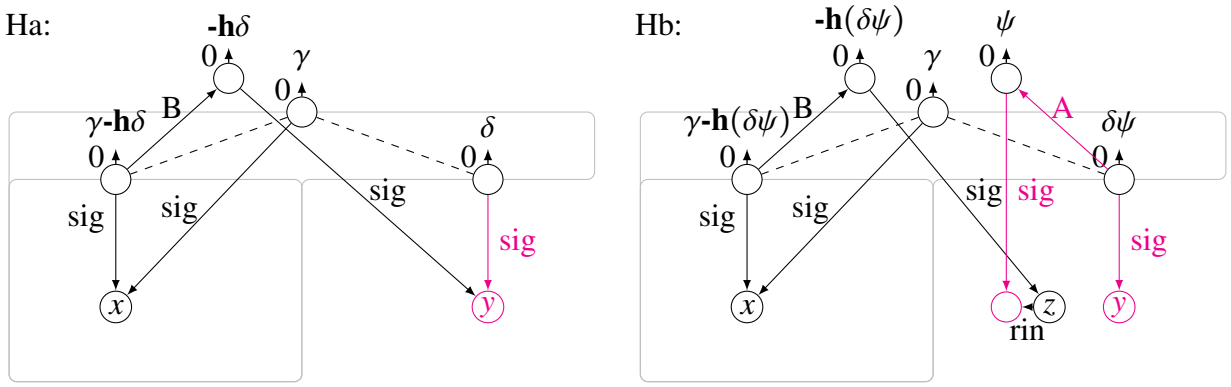


Figure 26: Results of grammatical inference rules for attachment of heavy-shifted or postposed signs.

Postposing of simple heavy signs. Signs containing no non-local dependencies can be posposed directly:

(81) Kim [$_{\textbf{V-aN}}$ [$_{\textbf{V-aN-hN}}$ learned _ in one day] [$_{\textbf{N}}$ the entire Ada programming language]].

In such cases, a sign of type γ-**h**δ containing a non-local dependency **-h**δ (where δ does not itself contain a non-local dependency) can combine with a postposed sign of type δ to form a sign of type γ. The signified state of the postposed sign is then the signified state y of the non-local dependency, and the signified state of the resulting sign is then the signified state x of the sign with the heavy-shifted dependency:

$$\lambda_{g:(\gamma\text{-}\mathbf{h}\delta\to\text{-}\mathbf{h}\delta\to\Delta)\to\Gamma}\ \lambda_{q:\varepsilon}\ \lambda_{h:\gamma\to\Delta} \\ (g\,(\lambda_p\,\lambda_{q'}\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),(q'\,y))))\!:\!\Gamma\ \in\ R_2 \qquad \text{(Ha)}$$

Cued associations resulting from this inference rule are shown in Figure 26a.

Postposing of heavy signs containing non-local dependencise. Signs containing non-local dependencies, such as relative and comparative clauses, can also be postposed:

(82) a. [$_{\textbf{V-h(C-rN)}}$ [$_{\textbf{N-h(C-rN)}}$ Everything _] stops] [$_{\textbf{C-rN}}$ that _ starts].
b. [$_{\textbf{N-h(Cthan-g(A-aN))}}$ Every [$_{\textbf{A-aN-h(Cthan-g(A-aN))}}$ taller _] house] [$_{\textbf{Cthan-g(A-aN)}}$ than my house is wide] is gone.

In such cases, a sign of type γ-**h**$(\delta\psi)$ containing a non-local dependency **-h**$(\delta\psi)$ (where ψ is another non-local dependency) can combine with a postposed sign of type $\delta\psi$ to form a sign of type γ. The signified state of the new non-local sign of type ψ is associated from the signified state of the old non-local sign z through a restriction inheritance ('rin'), and again, the signified state of the resulting sign is then the signified state of the sign with the heavy-shifted dependency:

$$\lambda_{g:(\gamma\text{-}\mathbf{h}(\delta\psi)\to\text{-}\mathbf{h}(\delta\psi)\to\Delta)\to\Gamma}\ \lambda_{q:\varepsilon}\ \lambda_{q'':\psi}\ \lambda_{h:\gamma\to\Delta} \\ (g\,(\lambda_p\,\lambda_{q'}\,h\,(\lambda_x\,\exists_y\,\exists_z\,(p\,x),(q\,y),(q'\,z),(q''\,(\mathbf{f}_{\mathbf{rin}}\,z)))))\!:\!\Gamma\ \in\ R_2 \qquad \text{(Hb)}$$

Cued associations resulting from this inference rule are shown in Figure 26b. A sample derivation of the sentence *Everything stops that starts* is shown in Figure 27.

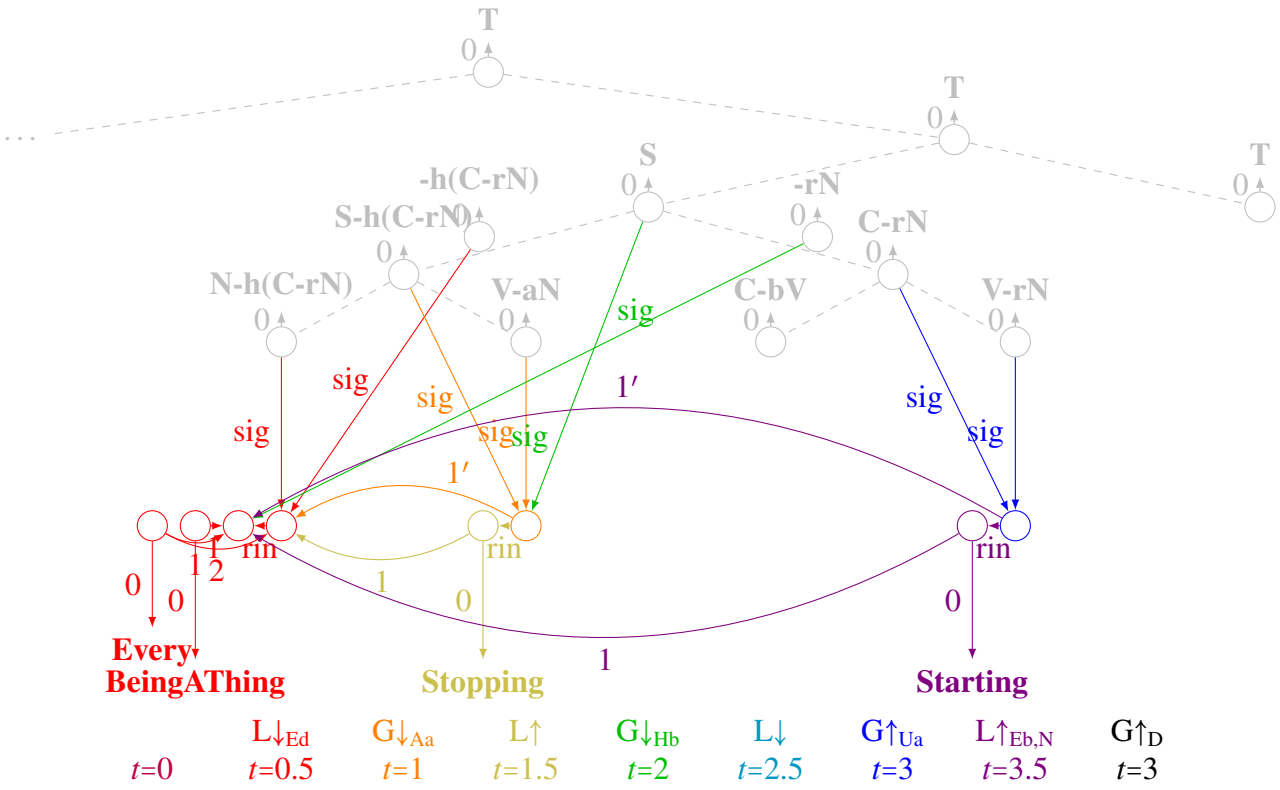


Figure 27: Cued associations derived from the sentence *Everything stops that starts*.

6.13 Passive attachment (unary)

Passive constructions (participial verb phrases **L-aN-vN** with unspecified subjects and constrained non-local dependencies) can be substituted for or coordinated with any predicative verb phrase **A-aN**:

(83) That bridge was [$_{\textbf{A-aN}}$ [$_{\textbf{L-aN-vN}}$ slept under _]].

This substitutability can be modeled as a unary grammatical inference rule which introduces a non-local sign of type **-vN** and identifies the signified state of this non-local sign with the first syntactic argument of signified state of the predicative verb phrase:

$$\lambda_{g:(\textbf{L-aN-vN}\rightarrow\textbf{-vN}\rightarrow\Delta)\rightarrow\Gamma}\ \lambda_{h:\textbf{A-aN}\rightarrow\Delta} \\ (g\,(\lambda_p\,\lambda_{q'}\,h\,(\lambda_y\,\exists_x\,(p\,x),(\mathbf{f_{ein}}\circ\mathbf{f_{rin}}\,y){=}x,(q'\,(\mathbf{f_{1'}}\,y))))):\Gamma\ \in\ R_1 \qquad \text{(V)}$$

Cued associations resulting from this inference rule are shown in Figure 28.

6.14 Zero-head introduction (unary)

Head words are sometimes elided (removed) from certain kinds of phrases. This can be modeled using a set of unary grammatical inference rules that replace elided function words or predicates.

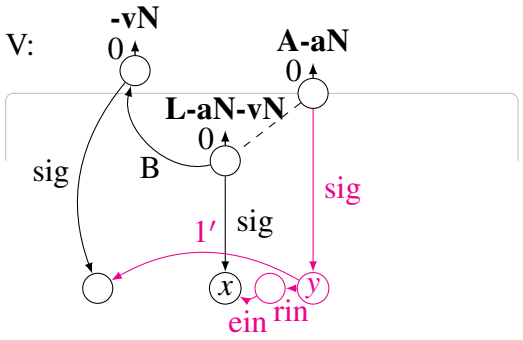


Figure 28: Results of grammatical inference rules for transforming passive verb phrases into predicative phrases.

Predicative noun phrases and appositives. Noun phrases (**N**) can often be substituted and coordinated with predicative phrases (**A-aN**), forming predicative noun phrases (in post-copular contextst) and appositives (in post-nominal modifier contexts):

(84) a. Ada is [$_{\mathbf{A\text{-}aN}}$ [$_{\mathbf{N}}$ an object-oriented programming language]].
b. Ada, [$_{\mathbf{A\text{-}aN}}$ [$_{\mathbf{N}}$ an object-oriented programming language]], was written in 1983.

These constructions can be modeled using unary grammatical inference rules which associate the first argument of the predicative phrase with the restrictor of the restrictor the referential state signified by the noun phrase:[6]

$$\lambda_{g:(\mathbf{N}\to\Delta)\to\Gamma}\ \lambda_{h:\mathbf{A\text{-}aN}\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(\mathbf{f}_{\mathbf{1}'}\,y)=(\mathbf{f}_{\mathbf{rin}}\circ\mathbf{f}_{\mathbf{rin}}\,x))))\!:\!\Gamma\ \in\ R_1 \qquad \text{(Za)}$$

Cued associations resulting from this inference rule are shown in Figure 29a.

Time and measure noun phrases and noun modifiers. In many cases, noun modifiers are connected to modificands by some elementary predication. For example, noun phrases *one centimeter*, *the week before Pat moved*, and *computer* in Examples 85a, 85b and 85c, serve as second participants of elementary predications **HavingMeasure**, **BeingDuring**, and **BeingFor**, whose first participants are the first syntactic arguments of the resulting modifier phrases:

(85) a. Open the door [$_{\mathbf{R\text{-}aN}}$ [$_{\mathbf{N}}$ one centimeter]].
b. Kim traveled [$_{\mathbf{R\text{-}aN}}$ [$_{\mathbf{N}}$ the week before Pat moved]].
c. Ada is a [$_{\mathbf{A\text{-}aN}}$ [$_{\mathbf{N}}$ computer]] language.

These constructions can be modeled with predicate-specific grammatical inference rules that add an elementary predication of type κ, whose first participant is the first syntactic argument of the resulting sign and whose second participant is the signified state of the noun phrase.

$$\begin{aligned}&\lambda_{g:(\mathbf{N}\to\Delta)\to\Gamma}\ \lambda_{h:\tau\text{-}\mathbf{a}\upsilon\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),\\ &\quad(\mathbf{f}_{\mathbf{0}}\circ\mathbf{f}_{\mathbf{rin}}\,y)=\kappa,(\mathbf{f}_{\mathbf{1}}\circ\mathbf{f}_{\mathbf{rin}}\,y)=(\mathbf{f}_{\mathbf{1}'}\,y),(\mathbf{f}_{\mathbf{2}}\circ\mathbf{f}_{\mathbf{rin}}\,y)=x))):\Gamma\ \in\ R_1\end{aligned} \qquad \text{(Zb-}\kappa\text{)}$$

Cued associations resulting from this inference rule are shown in Figure 29b.

[6]Double restriction inheritance associations are necessary to prevent participant collisions when the noun phase is a deverbal nominalization (with numbered participants) and the subject of the predicative phrase is a modificand and is also a deverbal nominalization (with numbered participants).

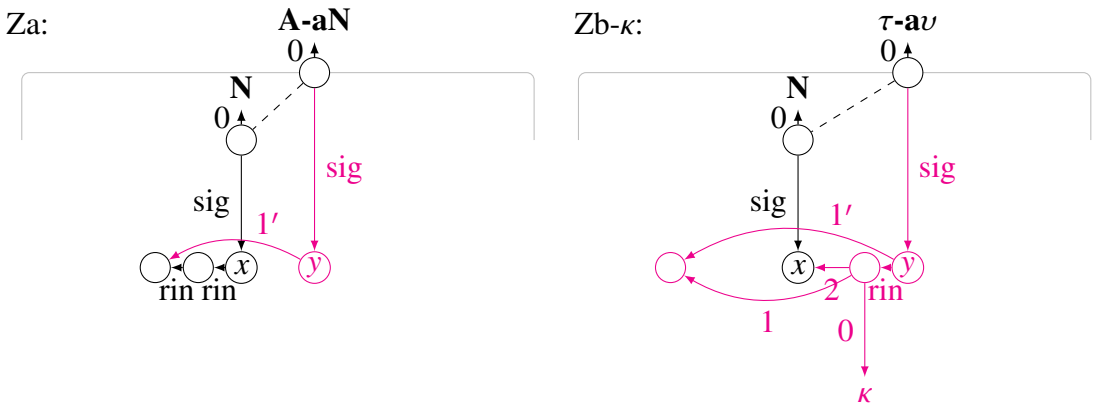


Figure 29: Results of grammatical inference rules for transforming passive verb phrases into predicative phrases.

References

Abney, S. P. and Johnson, M. (1991). Memory requirements and local ambiguities of parsing strategies. *J. Psycholinguistic Research*, 20(3):233–250.

Ajdukiewicz, K. (1935). Die syntaktische konnexitat. In McCall, S., editor, *Polish Logic 1920-1939*, pages 207–231. Oxford University Press. Translated from Studia Philosophica 1: 1–27.

Anderson, J. A., Silverstein, J. W., Ritz, S. A., and Jones, R. S. (1977). Distinctive features, categorical perception and probability learning: Some applications of a neural model. *Psychological Review*, 84:413–451.

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

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Barwise, J. and Cooper, R. (1981). Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4.

Copestake, A., Flickinger, D., Pollard, C., and Sag, I. (2005). Minimal recursion semantics: An introduction. *Research on Language and Computation*, pages 281–332.

Crocker, M. W. and Keller, F. (2005). Probabilistic grammars as models of gradience in language processing. In Fanselow, G., Féry, C., Vogel, R., and Schlesewsky, M., editors, *GRADIENCE IN GRAMMAR: GENERATIVE PERSPECTIVES*. University Press.

Davidson, D. (1967). The logical form of action sentences. In Rescher, N., editor, *The logic of decision and action*, pages 81–94. University of Pittsburgh Press, Pittsburgh.

Featherston, S. (2005). Universals and grammaticality: wh-constraints in German and English. *Linguistics*, 43(4):667–711.

Gibson, E. (1991). *A computational theory of human linguistic processing: Memory limitations and processing breakdown*. PhD thesis, Carnegie Mellon.

Hale, J. (2001). A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American chapter of the Association for Computational Linguistics*, pages 159–166, Pittsburgh, PA.

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

Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness*. Harvard University Press, Cambridge, MA, USA.

Karttunen, L. (1976). Discourse referents. In McCawley, J. D., editor, *Notes from the Linguistic Underground (Syntax and Semantics, vol. 7)*. Academic Press, New York.

Levy, R. (2008). Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Lewis, R. L. and Vasishth, S. (2005). An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, 29(3):375–419.

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

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (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.

Mel'čuk, I. (1988). *Dependency syntax: theory and practice*. State University of NY Press, Albany.

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

Nguyen, L., van Schijndel, M., and Schuler, W. (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.

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

Parsons, T. (1990). *Events in the Semantics of English*. MIT Press.

Rasmussen, N. E. and Schuler, W. (2017). Left-corner parsing with distributed associative memory produces surprisal and locality effects. *Cognitive Science*.

Resnik, P. (1992). Left-corner parsing and psychological plausibility. In *Proceedings of COLING*, pages 191–197, Nantes, France.

Rosenkrantz, S. J. and Lewis, II, P. M. (1970). Deterministic left corner parser. In *IEEE Conference Record of the 11th Annual Symposium on Switching and Automata*, pages 139–152.

Saussure, F. d. (1916). *Cours de Linguistique Générale*. Payot.

Schuler, W. and Wheeler, A. (2014). Cognitive compositional semantics using continuation dependencies. In *Third Joint Conference on Lexical and Computational Semantics (*SEM'14)*.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216.

Stabler, E. (1994). The finite connectivity of linguistic structure. In *Perspectives on Sentence Processing*, pages 303–336. Lawrence Erlbaum.

van Schijndel, M., Exley, A., and Schuler, W. (2013). A model of language processing as hierarchic sequential prediction. *Topics in Cognitive Science*, 5(3):522–540.

van Schijndel, M. and Schuler, W. (2013). An analysis of frequency- and memory-based processing costs. In *Proceedings of NAACL-HLT 2013*. Association for Computational Linguistics.

A Translating cued-association structures to lambda calculus

The cued association graphs above can be translated into lambda calculus (not a cognitive process).

1. Add a lambda term to Γ for each predication in Γ with no outscoped variables or inheritances:

$$\frac{\Gamma, (f\ x_0\ ..\ x_n\ ..\ x_N)\ ;\ \Delta}{\Gamma, (\lambda_{z_n}\ \tau_f\ z_0\ ..\ z_n\ ..\ z_N)\ ;\ \Delta}\ \overbrace{f \notin Q}^{\text{not quant}},\ \overbrace{\forall_y\ (\mathbf{f_{scope}}\ y){=}x_n \notin \Gamma}^{\text{no incoming scope}},\ \overbrace{\forall_y\ \forall_{f \in \{\mathbf{f_{rin}}, \mathbf{f_{cin}}, \mathbf{f_{uin}}\}}\ (f\ x_n){=}y \notin \Gamma}^{\text{no outgoing inheritance}} \quad \text{(P)}$$

2. Conjoin lambda terms over the same variable in Γ (this combines modifier predications):

$$\frac{\Gamma, (\lambda_u\ \phi), (\lambda_u\ \psi)\ ;\ \Delta}{\Gamma, (\lambda_u\ \phi \wedge \psi)\ ;\ \Delta} \quad \text{(C)}$$

3. Move terms in Γ with no missing predications, outscoped variables or inheritances to Δ:

$$\frac{\Gamma, (\lambda_{z_n}\ \psi)\ ;\ \Delta}{\Gamma\ ;\ (\lambda_{z_n}\ \psi), \Delta}\ \overbrace{\forall_{f' \notin Q}\ (f'..\ x_n\ ..) \notin \Gamma}^{\text{no elem. pred. in rest of graph}},\ \overbrace{\forall_y\ (\mathbf{f_{scope}}\ y){=}x_n \notin \Gamma}^{\text{no incoming scope}},\ \overbrace{\forall_y\ \forall_{f \in \{\mathbf{f_{rin}}, \mathbf{f_{cin}}, \mathbf{f_{uin}}\}}\ (f\ x_n){=}y \notin \Gamma}^{\text{no outgoing inheritance}} \quad \text{(M)}$$

4. Add translations τ_f of quantifiers f in Γ over complete lambda terms in Δ:

$$\frac{\Gamma, (f\ y\ x_n\ x_o)\ ;(\lambda_{z_n}\ \phi), (\lambda_{z_o}\ \psi), \Delta}{\Gamma, (\tau_f\ (\lambda_{z_n}\ \phi)\ (\lambda_{z_o}\ \psi))\ ;(\lambda_{z_n}\ \phi), (\lambda_{z_o}\ \psi), \Delta}\ f \in Q,\ \overbrace{\forall_{f' \in \{\mathbf{f_{*in}}\}}\ (f'..){=}x_o \notin \Gamma}^{\text{no inheritance arriving at nuc. scope}},\ \overbrace{\forall_{f' \in Q}\ (f'..\ x_o\ ..) \notin \Gamma}^{\text{no quant over n. s. in rest of graph}} \quad \text{(Q)}$$

5. Add lambda for outgoing scope:

$$\frac{\Gamma, (\mathbf{f_{scope}}\ x_o){=}x_m, (\tau_f\ (\lambda_{z_n}\ \phi)\ (\lambda_{z_o}\ \psi))\ ;\ \Delta}{\Gamma, (\lambda_{z_m}\ \tau_f\ (\lambda_{z_n}\ \phi)\ (\lambda_{z_o}\ \psi))\ ;\ \Delta}\ \overbrace{\forall_y\ (\mathbf{f_{scope}}\ z_n){=}y \notin \Gamma}^{\text{no scope departing restrictor}} \tag{S1}$$

$$\frac{\Gamma, (\mathbf{f_{scope}}\ x_n){=}x_\ell, (\mathbf{f_{scope}}\ x_o){=}x_m, (\tau_f\ (\lambda_{z_n}\ \phi)\ (\lambda_{z_o}\ \psi))\ ;\ \Delta}{\Gamma, (\mathbf{f_{scope}}\ x_n){=}x_\ell, (\lambda_{z_m}\ \tau_f\ (\lambda_{z_n}\ \underbrace{(\lambda_{z_\ell}\ \phi)\ z_m}_{\text{replace } z_\ell \text{ with } z_m})\ (\lambda_{z_o}\ \psi))\ ;\ \Delta} \tag{S2}$$

6. Add a lambda term to Γ for each inheritance that is empty or from complete term in Δ:

$$\frac{\Gamma, (f\ y){=}x_n\ ;\ \Delta}{\Gamma, (f\ y){=}x_n, \underbrace{(\lambda_{z_n}\ \mathbf{True})}_{\text{empty inheritance}}\ ;\ \Delta}\ f \in \{\mathbf{f_{rin}}, \mathbf{f_{cin}}\},\ \underbrace{\forall_{f' \notin Q}\ (f'..\ x_n\ ..) \notin \Gamma}_{\text{no elem. pred. on } x \text{ in rest of graph}} \tag{I1}$$

$$\frac{\Gamma, (f\ x_m){=}x_n\ ;\ (\lambda_{z_n}\ \phi), \Delta}{\Gamma, (\lambda_{z_m}\ \underbrace{(\lambda_{z_n}\ \phi)\ z_m}_{\text{replace inherited with inheritor}})\ ;\ (\lambda_{z_n}\ \phi), \Delta}\ f \in \{\mathbf{f_{rin}}, \mathbf{f_{cin}}\},\ \underbrace{\forall_y\ (\mathbf{f_{scope}}\ x_n){=}y \notin \Gamma}_{\text{no scope departing inherited}} \tag{I2}$$

$$\frac{\Gamma, (\mathbf{f_{scope}}\ x_n){=}y, (f\ x_o){=}x_n\ ;\ (\lambda_{z_n}\ \phi), \Delta}{\Gamma, (\mathbf{f_{scope}}\ x_n){=}y, \underbrace{(\mathbf{f_{scope}}\ x_o){=}y}_{\text{inherit scope}}, (\lambda_{z_o}\ \underbrace{(\lambda_{z_n}\ \phi)\ z_o}_{\text{replace inherited with inheritor}})\ ;\ (\lambda_{z_n}\ \phi), \Delta}\ f \in \{\mathbf{f_{rin}}, \mathbf{f_{cin}}\},\ \underbrace{\forall_z (\mathbf{f_{scope}}\ x_o){=}z \notin \Gamma}_{\text{no scope departing inheritor}} \tag{I3}$$

$$\frac{\Gamma, (\mathbf{f_{scope}}\ x_n){=}x_\ell, (\mathbf{f_{scope}}\ x_o){=}x_m, (f\ x_o){=}x_n\ ;\ (\lambda_{z_n}\ \phi), \Delta}{\Gamma, (\mathbf{f_{scope}}\ x_n){=}x_\ell, (\mathbf{f_{scope}}\ x_o){=}x_m, (\lambda_{z_o}\ (\lambda_{z_\ell}\ \underbrace{(\lambda_{z_n}\ \phi)\ z_o}_{\text{replace inherited with inheritor}})\ z_m)\ ;\ (\lambda_{z_n}\ \phi), \Delta}\ f \in \{\mathbf{f_{rin}}, \mathbf{f_{cin}}\} \tag{I4}$$

replace scope of inherited with scope of inheritor

7. Add union of conjunct constraints to complete lambda terms in Δ:

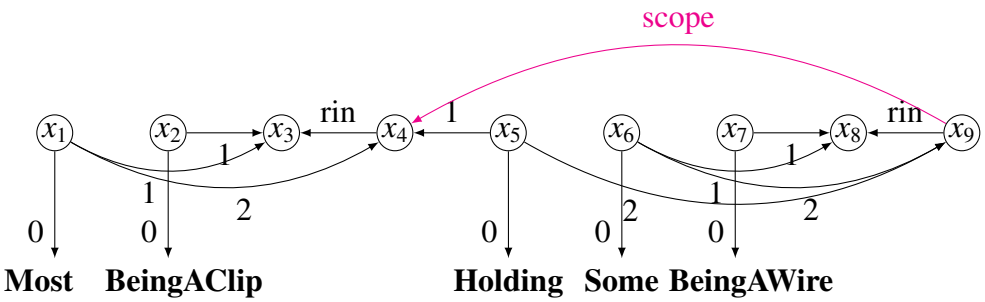
$$\frac{\Gamma, (\mathbf{f_{uin}}\ y){=}x\ ;\ \Delta}{\Gamma, (\mathbf{f_{cin}}\ y){=}x, (\mathbf{f_{union}}\ y){=}x\ ;\ \Delta} \tag{U1}$$

$$\frac{\Gamma, (\mathbf{f_{union}}\ x_o){=}x_n\ ;\ (\lambda_{z_o}\ \psi), \Delta}{\Gamma, (\lambda_{z_n}\ (\lambda_{z_o}\ \psi)\ z_n)\ ;\ (\lambda_{z_o}\ \psi), \Delta}\ (\lambda_{z_n}\ ...) \notin \Gamma \tag{U2}$$

$$\frac{\Gamma, (\lambda_{z_n}\ \phi), (\mathbf{f_{union}}\ x_o){=}x_n\ ;\ (\lambda_{z_o}\ \psi), \Delta}{\Gamma, (\lambda_{z_n}\ \phi \vee (\lambda_{z_o}\ \psi)\ z_n)\ ;\ (\lambda_{z_o}\ \psi), \Delta} \tag{U3}$$

$$\frac{\Gamma, (\lambda_{z_n}\ \phi)\ ;\ (\lambda_{z_n}\ \psi), \Delta}{\Gamma\ ;\ (\lambda_{z_n}\ \phi \wedge \psi), \Delta}\ \forall_y\ (\mathbf{f_{union}}\ y){=}x_n \notin \Gamma \tag{U4}$$

For example, our graph:



translates into:

(**Most** x_1 x_3 x_4), (**f**$_{\textbf{rin}}$ x_4)=x_3, (**f**$_{\textbf{scope}}$ x_9)=x_4, (**Some** x_6 x_8 x_9), (**f**$_{\textbf{rin}}$ x_9)=x_8, (**Holding** x_5 x_4 x_9), (**Clip** x_2 x_3), (**Wire** x_7 x_8) ;

P (**Most** x_1 x_3 x_4), (**f**$_{\textbf{rin}}$ x_4)=x_3, (**f**$_{\textbf{scope}}$ x_9)=x_4, (**Some** x_6 x_8 x_9), (**f**$_{\textbf{rin}}$ x_9)=x_8, (**Holding** x_5 x_4 x_9), (λ_{z_3} **Clip** z_2 z_3), (λ_{z_8} **Wire** z_7 z_8) ;

M (**Most** x_1 x_3 x_4), (**f**$_{\textbf{rin}}$ x_4)=x_3, (**f**$_{\textbf{scope}}$ x_9)=x_4, (**Some** x_6 x_8 x_9), (**f**$_{\textbf{rin}}$ x_9)=x_8, (**Holding** x_5 x_4 x_9) ; (λ_{z_3} **Clip** z_2 z_3), (λ_{z_8} **Wire** z_7 z_8)

*I*2 (**Most** x_1 x_3 x_4), (**f**$_{\textbf{rin}}$ x_4)=x_3, (**f**$_{\textbf{scope}}$ x_9)=x_4, (**Some** x_6 x_8 x_9), (λ_{z_9} **Wire** z_7 z_9), (**Holding** x_5 x_4 x_9) ; (λ_{z_3} **Clip** z_2 z_3), (λ_{z_8} **Wire** z_7 z_8)

P (**Most** x_1 x_3 x_4), (**f**$_{\textbf{rin}}$ x_4)=x_3, (**f**$_{\textbf{scope}}$ x_9)=x_4, (**Some** x_6 x_8 x_9), (λ_{z_9} **Wire** z_7 z_9), (λ_{z_9} **Holding** z_5 z_4 z_9) ; (λ_{z_3} **Clip** z_2 z_3), (λ_{z_8} **Wire** z_7 z_8)

C (**Most** x_1 x_3 x_4), (**f**$_{\textbf{rin}}$ x_4)=x_3, (**f**$_{\textbf{scope}}$ x_9)=x_4, (**Some** x_6 x_8 x_9), (λ_{z_9} **Wire** z_7 z_9 ∧ **Holding** z_5 z_4 z_9) ; (λ_{z_3} **Clip** z_2 z_3), (λ_{z_8} **Wire** z_7 z_8)

M (**Most** x_1 x_3 x_4), (**f**$_{\textbf{rin}}$ x_4)=x_3, (**f**$_{\textbf{scope}}$ x_9)=x_4, (**Some** x_6 x_8 x_9) ; (λ_{z_9} **Wire** z_7 z_9 ∧ **Holding** z_5 z_4 z_9), (λ_{z_3} **Clip** z_2 z_3), (λ_{z_8} **Wire** z_7 z_8)

Q (**Most** x_1 x_3 x_4), (**f**$_{\textbf{rin}}$ x_4)=x_3, (**f**$_{\textbf{scope}}$ x_9)=x_4, (**Some** (λ_{z_8} **Wire** z_7 z_8) (λ_{z_9} **Wire** z_7 z_9 ∧ **Holding** z_5 z_4 z_9)) ; (λ_{z_3} **Clip** z_2 z_3), (λ_{z_8} ...), (λ_{z_9} ...)

S 1 (**Most** x_1 x_3 x_4), (**f**$_{\textbf{rin}}$ x_4)=x_3, (λ_{z_4} **Some** (λ_{z_8} **Wire** z_7 z_8) (λ_{z_9} **Wire** z_7 z_9 ∧ **Holding** z_5 z_4 z_9)) ; (λ_{z_3} **Clip** z_2 z_3), (λ_{z_8} ...), (λ_{z_9} ...)

*I*2 (**Most** x_1 x_3 x_4), (λ_{z_4} **Clip** z_2 z_4), (λ_{z_4} **Some** (λ_{z_8} **Wire** z_7 z_8) (λ_{z_9} **Wire** z_7 z_9 ∧ **Holding** z_5 z_4 z_9)) ; (λ_{z_3} ...), (λ_{z_8} ...), (λ_{z_9} ...)

C (**Most** x_1 x_3 x_4), (λ_{z_4} **Clip** z_2 z_4 ∧ **Some** (λ_{z_8} **Wire** z_7 z_8) (λ_{z_9} **Wire** z_7 z_9 ∧ **Holding** z_5 z_4 z_9)) ; (λ_{z_3} ...), (λ_{z_8} ...), (λ_{z_9} ...)

M (**Most** x_1 x_3 x_4) ; (λ_{z_4} **Clip** z_2 z_4 ∧ **Some** (λ_{z_8} **Wire** z_7 z_8) (λ_{z_9} **Wire** z_7 z_9 ∧ **Holding** z_5 z_4 z_9)), (λ_{z_3} ...), (λ_{z_8} ...), (λ_{z_9} ...)

Q (**Most** (λ_{z_3} **Clip** z_2 z_3) (λ_{z_4} **Clip** z_2 z_4 ∧ **Some** (λ_{z_8} **Wire** z_7 z_8) (λ_{z_9} **Wire** z_7 z_9 ∧ **Holding** z_5 z_4 z_9))) ; (λ_{z_3} ...), (λ_{z_8} ...), (λ_{z_9} ...)

B Translating store functions to cued-association structures

Translation from store functions to cued-association structures depends on the type of the store:

$$T\,(g:\underbrace{\alpha}_{\text{top sign}}) = (\lambda_a\ \underbrace{(\mathbf{f_0}\,a)=\mathbf{x}_\alpha}_{\text{const for top sign}}, \underbrace{(g\,(\mathbf{f_{sig}}\,a))}_{\text{store fn applied at signified state of top sign}}) \qquad \text{(Base Case)}$$

$$T\,(g:\underbrace{\beta}_{\text{expected sign}}\to\Gamma) = (\lambda_b\ \underbrace{(\mathbf{f_0}\,b)=\mathbf{x}_\beta}_{\text{const for expected sign}}, \underbrace{((T\,(g\ \underbrace{(\lambda_x\,(\mathbf{f_{sig}}\,b)=x)}_{\text{property of sig state of exp sign}}))\ \underbrace{(\mathbf{f_A}\,b)}_{\text{sign after expected sign}})}_{\text{cued-assoc struct of store fn after expected sign}}) \qquad \text{(L. Child)}$$

$$T\,(g:\underbrace{(\alpha\to\Delta)}_{\text{hole between incomplete signs}}\to\Gamma) = (\lambda_a\ \underbrace{(\mathbf{f_0}\,a)=\mathbf{x}_\alpha}_{\text{const for beginning of hole}}, \underbrace{((T\,(\lambda_{h:\Delta}\,g\ \underbrace{(\lambda_{p:\alpha}\,h,(p\,(\mathbf{f_{sig}}\,a))))}_{\text{original hole}}))\ \underbrace{(\mathbf{f_B}\,a)}_{\text{sign after beginning of hole}})}_{\text{cued-assoc struct of store fn after beginning of hole}}) \qquad \text{(R. Child)}$$

We now represent the syntax of partial analyses as stores (and words as units with unit subtypes):

$$g:\beta\to\overbrace{(\alpha\to\beta')}^{\text{hole between incomplete signs}}\to(\alpha'\to\beta'')\to\ldots\to\alpha''\ \cdot\ \mathbf{unit}:\omega_1\ \cdot\ \mathbf{unit}:\omega_2\ \cdot\ \mathbf{unit}:\omega_3\ \cdots$$

For example:

$$g : \mathbf{N-aD} \rightarrow (\mathbf{N} \rightarrow \mathbf{T}) \rightarrow \mathbf{T} \;\cdot\; \mathbf{unit} : \omega_1 \;\cdot\; \mathbf{unit} : \omega_2 \;\cdot\; \mathbf{unit} : \omega_3 \;\cdots$$

Derivation of translation:

$T(g : \mathbf{N-aD} \rightarrow (\mathbf{N} \rightarrow \mathbf{T}) \rightarrow \mathbf{T})$

R.C. $\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{N-aD}, ((T\,(g\,(\lambda_x\,(\mathbf{f_{sig}}\,b){=}x)) : (\mathbf{N} \rightarrow \mathbf{T}) \rightarrow \mathbf{T})\,(\mathbf{f_A}\,b))$

L.C. $\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{N-aD}, ((\lambda_a\,(\mathbf{f_0}\,a){=}\mathbf{N}, ((T\,(\lambda_h\,(g\,(\lambda_x\,(\mathbf{f_{sig}}\,b){=}x)\,(\lambda_p\,h, (p\,(\mathbf{f_{sig}}\,a)))) : \mathbf{T} \rightarrow \mathbf{T}))\,(\mathbf{f_B}\,a)))(\mathbf{f_A}\,b))$

reduce $\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{N-aD}, (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\mathbf{N}, ((T\,(\lambda_h\,(g\,(\lambda_x\,(\mathbf{f_{sig}}\,b){=}x)\,(\lambda_p\,h, (p\,(\mathbf{f_{sig}}{\circ}\mathbf{f_A}\,b)))) : \mathbf{T} \rightarrow \mathbf{T}))\,(\mathbf{f_B}{\circ}\mathbf{f_A}\,b))$

R.C. $\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{N-aD}, (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\mathbf{N}, ((\lambda_{b'}\,(\mathbf{f_0}\,b'){=}\mathbf{T}, ((T\,((\lambda_h\,(g\,(\lambda_x\,(\mathbf{f_{sig}}\,b){=}x)\,(\lambda_p\,h, (p\,(\mathbf{f_{sig}}{\circ}\mathbf{f_A}\,b))))))\,(\lambda_x\,(\mathbf{f_{sig}}\,b'){=}x)) : \mathbf{T})\,(\mathbf{f_A}\,b')))(\mathbf{f_B}{\circ}\mathbf{f_A}\,b))$

reduce $\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{N-aD}, (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\mathbf{N}, ((\lambda_{b'}\,(\mathbf{f_0}\,b'){=}\mathbf{T}, ((T\,(g\,(\lambda_x\,(\mathbf{f_{sig}}\,b){=}x)\,(\lambda_p\,\lambda_x\,(\mathbf{f_{sig}}\,b'){=}x, (p\,(\mathbf{f_{sig}}{\circ}\mathbf{f_A}\,b)))) : \mathbf{T})\,(\mathbf{f_A}\,b')))(\mathbf{f_B}{\circ}\mathbf{f_A}\,b))$

reduce $\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{N-aD}, (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\mathbf{N}, (\mathbf{f_0}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\mathbf{T}, ((T\,(g\,(\lambda_x\,(\mathbf{f_{sig}}\,b){=}x)\,(\lambda_p\,\lambda_x\,(\mathbf{f_{sig}}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}x, (p\,(\mathbf{f_{sig}}{\circ}\mathbf{f_A}\,b)))) : \mathbf{T})\,(\mathbf{f_A}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b))$

B.C. $\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{N-aD}, (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\mathbf{N}, (\mathbf{f_0}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\mathbf{T}, ((\lambda_a(\mathbf{f_0}\,a){=}\mathbf{T}, (g\,(\lambda_x\,(\mathbf{f_{sig}}\,b){=}x)\,(\lambda_p\,\lambda_x\,(\mathbf{f_{sig}}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}x, (p\,(\mathbf{f_{sig}}{\circ}\mathbf{f_A}\,b)))\,(\mathbf{f_{sig}}\,a)) : \mathbf{T})\,(\mathbf{f_A}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b))$

reduce $\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{N-aD}, (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\mathbf{N}, (\mathbf{f_0}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\mathbf{T}, (\mathbf{f_0}{\circ}\mathbf{f_A}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\mathbf{T}, (g\,(\lambda_x\,(\mathbf{f_{sig}}\,b){=}x)\,(\lambda_p\,\lambda_x\,(\mathbf{f_{sig}}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}x, (p\,(\mathbf{f_{sig}}{\circ}\mathbf{f_A}\,b)))\,(\mathbf{f_{sig}}{\circ}\mathbf{f_A}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b)) : \mathbf{T}$

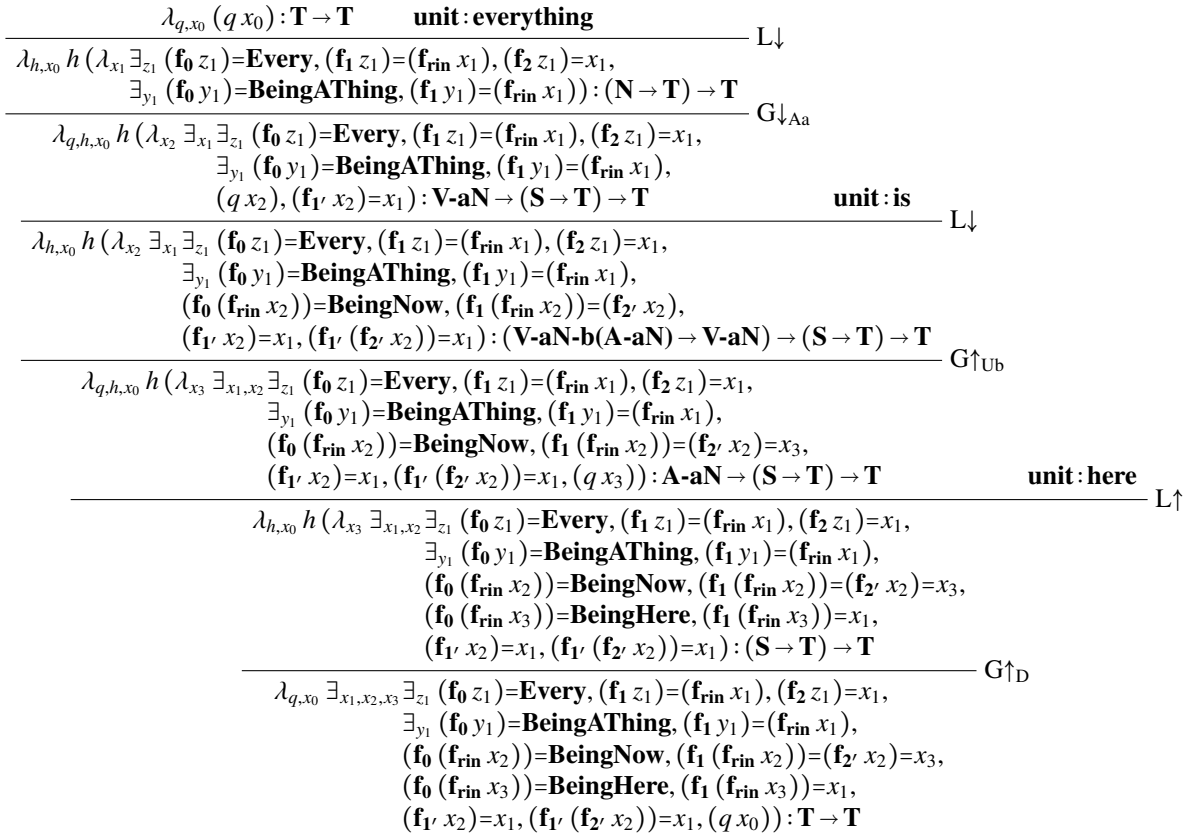
This will simplify our grammatical inference rules, defined in the next section.

C Derivations of example sentences

Cued-association structures can be derived using these grammatical inference rules in applications L↓, L↑, G↓, and G↑ as defined in Section 3. These derivations start with an empty store of type $\mathbf{T} \rightarrow \mathbf{T}$ followed by a sequence of unit tokens with word types, and alternately apply lexical inference rules and grammatical inference rules to consume all the words in the sequence and produce another store of type $\mathbf{T} \rightarrow \mathbf{T}$ containing a complete corresponding cued-association structure.

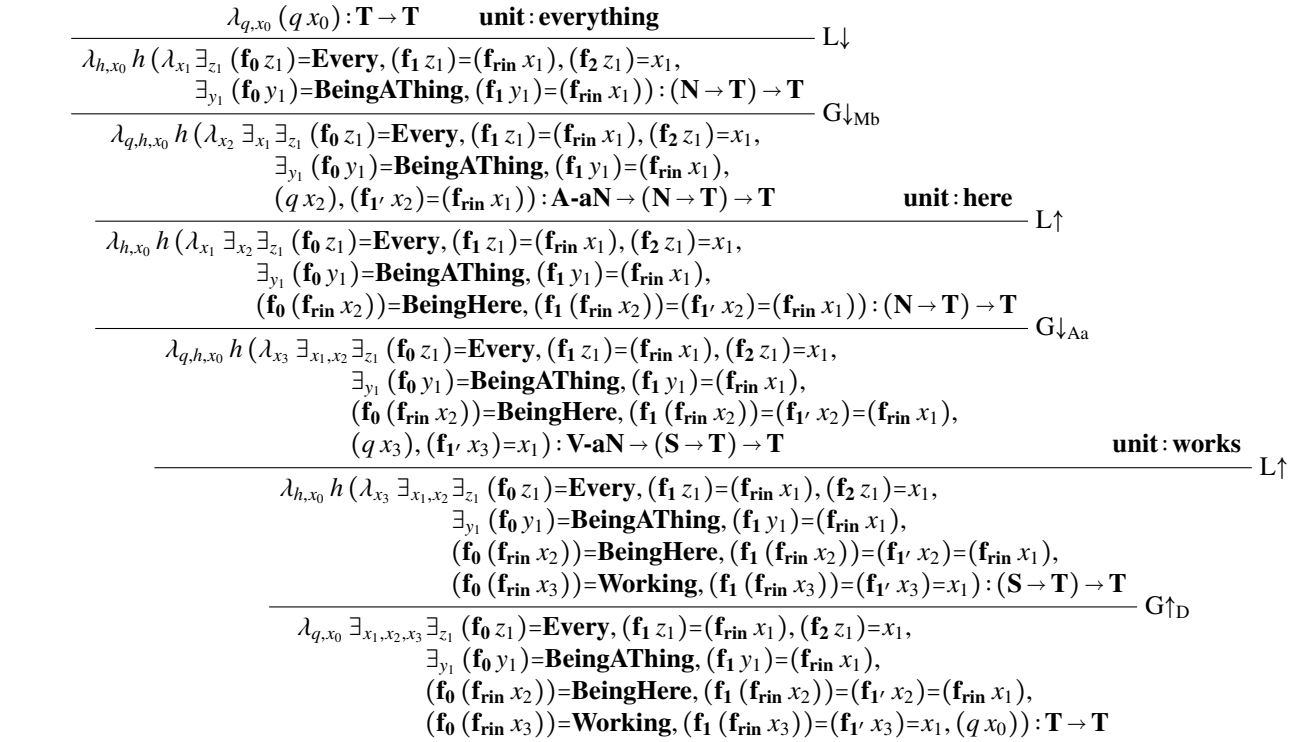
C.1 Auxiliary attachment.

Below is a derivation of the sentence *Everything is here,* shown in Figure 12 in Section 6.3:



C.2 Modifier attachment.

Below is a derivation of the sentence *Everything here works,* shown in Figure 13 in Section 6.4:



C.3 Gap-filler attachment.

Below is a derivation of the sentence *Everything here someone likes,* shown in Figure 21 in Section 6.8: