Ling 8700: Lecture Notes 4
From Cued Association Semantics to Sentence Processing

So far, we have introduced a model of **complex ideas**, consisting of...

- **referential states**: countable, distinguishable entities

  (at algorithmic level, modeled as **vectors** of activation in attentional focus)

  (at computational level, notated as entities u, v, w, x, y, z)

- **cued associations**: labeled relations between entities

  (at algorithmic level, modeled as **matrices** of synaptic weights in associative memory)

  (at computational level, notated as functions f from referential states to referential states)

- these form **elementary predications**: referential states which have...

  1. **predication type constants**: information that defines the specific role of each participant.

     (at algorithmic level, modeled as characteristic features of patterns)

     (notated as association $\mathbf{f_0}$ from elementary predications to type constants $\mathbf{x}_\alpha$)

  2. **participants**: related referential states s.t. relation and predication have same extent

     (notated as numbered associations: $\mathbf{f_1}$, $\mathbf{f_2}$, ..., from elem. predications to participants)

- these form **cued association structures**: sets of cued associations in associative memory

  (notated as functions p, q, etc. from referential states to truth values)

These lecture notes describe a model of sentence processing as encoding complex ideas.

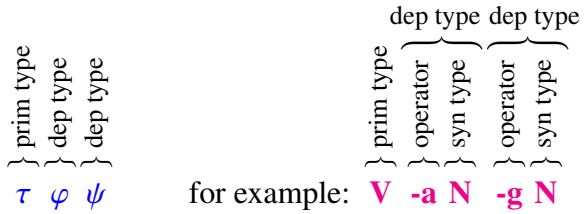
Contents

4.1 Typed signs

The sentence processing model described in these notes operates on **signs** (Saussure, 1916).

Signs are referential states for instances of words, phrases and clauses, with...

1. **signified referential states**: referential states described by signs
   (notated using $\mathbf{f_{sig}}$ associations from signs to signified referential states)
2. **hierarchic associations**: superordinate incomplete signs, to be completed later in processing
   (notated using $\mathbf{f_A}$ and $\mathbf{f_B}$ associations from signs to superordinate signs)
3. **syntactic type constants**: category information which helps disambiguate structural attachments
   (types are features, but the relationship between signs and types is notated using $\mathbf{f_0}$ functions)
   (type values are notated with variables α, β, γ, δ, and ε over domain S)
   Syntactic type constants consist of ... (Ajdukiewicz, 1935; Bar-Hillel, 1953; Oehrle, 1994)
   (a) a **primitive clausal type**: distinct types for complete clauses with no unsatisfied arguments
   (notated τ or υ over domain T)
   (b) zero or more **syntactic dependency types**: types for required argument/conjunct/... signs
   (notated φ or ψ over domain $O \times S$)
   Syntactic dependency types consist of...
      i. a **type-constructing operator** (e.g. argument, modifier, gap filler), in domain O
      ii. another **syntactic type constant** for the dependent sign, in domain S



Additionally **signified referential states** may have…

- **syntactic dependencies** (subject, direct object, etc.), corresponding to the type of the sign

  (notated as associations $\mathbf{f_{1'}}$, $\mathbf{f_{2'}}$, … from signified referential states to dependent referential states)

4.2 Syntactic types for English (Nguyen et al., 2012)

Here is a list of primitive clausal types for English (distinguished by non-substitutability):

V:	finite verb clause	(e.g. *She believes that* [$_{\mathbf{V}}$ *he knows the truth*].)
I:	infinitive clause	(e.g. *She allows* [$_{\mathbf{I}}$ *him to know the truth*].)
B:	base-form clause	(e.g. *She requires that* [$_{\mathbf{B}}$ *he know the truth*].)
L:	participial clause	(e.g. *They have* [$_{\mathbf{L\text{-}aN}}$ *known the truth*] — no complete clause)
A:	adjectival / predicative clause	(e.g. *She kept* [$_{\mathbf{A}}$ *him knowing the truth*].)
R:	adverbial clause	(e.g. *They do it* [$_{\mathbf{R\text{-}aN}}$ *knowingly*] — no complete clause)
G:	gerund clause	(e.g. *She works without* [$_{\mathbf{G}}$ *him knowing the truth*].)
P:	particle	(e.g. *She picked him* [$_{\mathbf{Pup}}$ *up*] — lexical item specified in type)
T:	top-level discourse	(used for most superordinate incomplete sign in parser)
S:	top-level utterance	(e.g. *She says:* [$_{\mathbf{S}}$ *know the truth*]*!*)
Q:	subject-auxiliary inverted	(e.g. *She asks:* [$_{\mathbf{Q}}$ *did he know the truth*]*?*)
C:	complementized finite verb	(e.g. *She fretted* [$_{\mathbf{C}}$ *that he knows the truth*].)
F:	complementized infinitive	(e.g. *She waits* [$_{\mathbf{F}}$ *for him to know the truth*].)
E:	complementized base-form	(e.g. *She requires* [$_{\mathbf{E}}$ *that he know the truth*].)
N:	nominal clause / noun phrase	(e.g. *She talks about* [$_{\mathbf{N}}$ *his knowledge of the truth*].)
D:	determiner / possesive	(e.g. *She calculates* [$_{\mathbf{D}}$ *his knowledge of the truth's*] *effect.*)
O:	non-possessive genitive	(e.g. *She tires* [$_{\mathbf{O}}$ *of his knowledge of the truth*].)

English type-constructing operators w. examples distinguishing **N**-op-**N** contexts:

-a,**-b**: for unsatisfied requirements of preceding/succeeding ordinary arguments

(distinguishing signs that look ahead from signs that look behind for ordinary arguments):

(1) a. *We find* [$_{\mathbf{A}}$ [$_{\mathbf{N}}$ *our hero*] [$_{\mathbf{A\text{-}aN}}$ [$_{\mathbf{N}}$ *a dollar*] [$_{\mathbf{A\text{-}aN\text{-}aN}}$ *short*]]].
b. *We find* [$_{\mathbf{A}}$ [$_{\mathbf{N}}$ *our hero*] [$_{\mathbf{A\text{-}aN}}$ [$_{\mathbf{A\text{-}aN\text{-}bN}}$ *with*] [$_{\mathbf{N}}$ *only two dollars*]]].

-c,**-d**: for unsatisfied requirements of preceding/succeeding conjunct arguments

(distinguishing conjunct requirements from argument requirements):

(2) a. *The spy stole* [$_{\text{N}}$ *files*] [$_{\text{N-cN}}$ *and documents*].
b. *The spy stole files* [$_{\text{N-cN-dN}}$ *and*] [$_{\text{N}}$ *documents*].

-g: for unsatisfied requirements of gap filler arguments

(distinguishing gap-filler requirements from argument or conjunct requirements):

(3) *These are the bridges the boss was* [$_{\text{A-aN-gN}}$ *expected to think _ were unsafe*].

-h: for unsatisfied requirements of extraposed or heavy shifted categories

(distinguishing extraposition and heavy shift from gap filler requirements):

(4) *An article was* [$_{\text{A-aN-hN}}$ *expected to condemn _*] *this week over 100 unsafe bridges.*

(note requirements for extraposition and heavy shift are more restrictive than for gap fillers):

(5) *? The boss was* [$_{\text{A-aN-hN}}$ *expected to think _ were unsafe*] *this week over 100 bridges.*

-v: for unsatisfied requirements of passivized arguments

(distinguishing passive argument requirements from gap filler and heavy shift requirements):

(6) a. *That bridge was slept* [$_{\text{R-aN-vN}}$ *under _*].
b. *? That bridge was considered* [$_{\text{A-vN}}$ *_ unsafe*].

(note that passivization is more restrictive than extraposition or heavy shift):

(7) *** That bridge was considered* [$_{\text{A-vN}}$ [$_{\text{N}}$ *the boss*] [$_{\text{A-aN-vN}}$ *a critic of _*]].

-i,**-r**: for unsatisfied requirements of interrogative or relative pronoun antecedents

(interrogative/relative pronoun antecedent requirements differ from ea. other & other deps):

(8) a. *The spies knew* [$_{\text{N-iN}}$ *what*] *they had to steal _.*
b. *The spies knew of a device* [$_{\text{N-rN}}$ *the plans for which*] *they could steal _.*

Combinations of these primitive categories and type-combining operators create complex categories which can define several subcategorization frames:

V-aN: intransitive (e.g. *Pat* [$_{\text{V-aN}}$ *stayed*].)
V-aN-bN: transitive (e.g. *Pat* [$_{\text{V-aN-bN}}$ *described*] *the plan.*)
V-aN-bN-bN: ditransitive (e.g. *Pat* [$_{\text{V-aN-bN-bN}}$ *gave*] *the chief the plans.*)
V-aN-bC: sentential complement (e.g. *Pat* [$_{\text{V-aN-bC}}$ *thought*] *that it rained.*)
V-aN-bC-bN: ditransitive sentential complement (e.g. *Pat* [$_{\text{V-aN-bC-bN}}$ *told*] *us that it rained.*)
V-aN-b(I-aN): subject control (e.g. *Pat* [$_{\text{V-aN-b(I-aN)}}$ *decided*] *to leave.*)
V-aN-b(L-aN): auxiliary (e.g. *Pat* [$_{\text{V-aN-b(L-aN)}}$ *has*] *waited.*)

These subcategorizations can be generalized across verb forms:

B-aN: intransitive (e.g. *We let Pat* [$_{\text{B-aN}}$ *stay*].)
B-aN-bN: transitive (e.g. *We let Pat* [$_{\text{B-aN-bN}}$ *describe*] *the plan.*)

B-aN-bN-bN: ditransitive (e.g. *We let Pat* $[_{\textbf{B-aN-bN-bN}}$ *give*$]$ *the chief the plans.*)
B-aN-bC: sentential complement (e.g. *We let Pat* $[_{\textbf{B-aN-bC}}$ *think*$]$ *that it rained.*)
B-aN-bC-bN: ditransitive sentential complement (e.g. *We let Pat* $[_{\textbf{B-aN-bC-bN}}$ *tell*$]$ *us that it rained.*)
B-aN-b(I-aN): subject control (e.g. *We let Pat* $[_{\textbf{B-aN-b(I-aN)}}$ *decide*$]$ *to leave.*)

and to nominalizations:

N-aD: intransitive (e.g. *We talked about Pat's* $[_{\textbf{N-aD}}$ *stay*$]$.)
N-aD-bO: transitive (e.g. *We talked about Pat's* $[_{\textbf{N-aD-bO}}$ *description*$]$ *of the plan.*)
N-aD-bC: sentential complement (e.g. *We talked about Pat's* $[_{\textbf{N-aD-bC}}$ *thought*$]$ *that it rained.*)
N-aD-b(I-aN): subject control (e.g. *We talked about Pat's* $[_{\textbf{N-aD-b(I-aN)}}$ *decision*$]$ *to leave.*)

This specification of argument categories also allows marker words to be defined:

C-bV: finite complementizer (e.g. *We thought* $[_{\textbf{C-bV}}$ *that*$]$ *it rained.*)
F-bI: infinitive complementizer (e.g. *We waited* $[_{\textbf{F-bI}}$ *for*$]$ *it to rain.*)
E-bB: base-form complementizer (e.g. *We required* $[_{\textbf{E-bB}}$ *that*$]$ *it rain.*)
O-bN: genitive marker (e.g. *We got tired* $[_{\textbf{O-bN}}$ *of*$]$ *the rain.*)
D-aN: possessive marker (e.g. *That was Pat* $[_{\textbf{D-aN}}$ *'s*$]$ *flight.*)

4.3 Prediction hierarchies

Semantic associations must be formed as each lexeme is encountered during comprehension.

Hierarchic associations may form at the same time, based on recurrent transitions (Schuler, 2014).

Hierarchic transitions also hypothesized for learning hierarchic plans (making tea; Botvinick, 2007).

Left-corner parsers (Rosenkrantz and Lewis, 1970; Johnson-Laird, 1983; Resnik, 1992) assume working memory stores a prediction hierarchy of nested contexts, starting w. most recent sign b.

Can represent as a **cued-association structure** p concatenated w. sequence of input words $w_1, w_2, ..$:

$$p \cdot w_1 \cdot w_2 \cdot w_3 \cdots \quad \text{where } p = (\lambda_b\,(\mathbf{f_0}\,b){=}\beta, (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\alpha, (\mathbf{f_0}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\beta', (\mathbf{f_0}{\circ}\mathbf{f_A}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\alpha', \ldots)$$

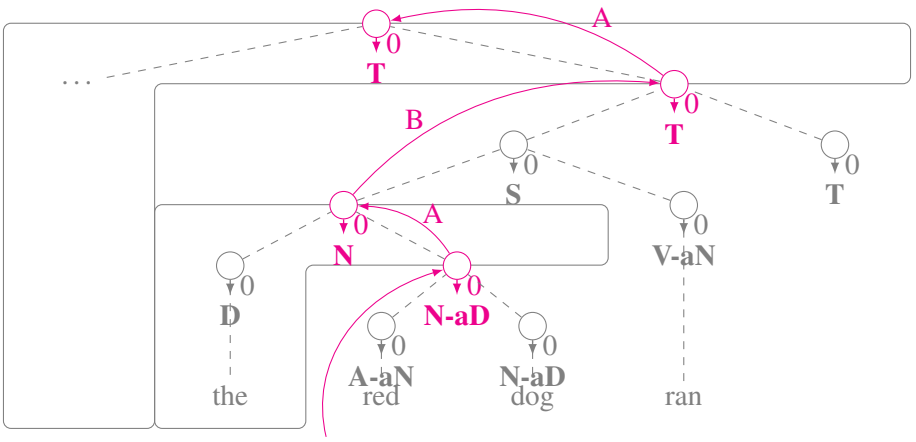
Between sentences, a parser may expect a discourse (**T**) to complete a discourse (**T**) as a default:

$$p \cdot w_1 \cdot w_2 \cdot w_3 \cdots \quad \text{where } p = (\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{T}, (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\mathbf{T})$$

After the word '*the,*' it may then expect a common noun (**N-aD**) to complete a noun phrase (**N**), within a discourse (**T**) to complete a discourse (**T**):

$$p \cdot w_1 \cdot w_2 \cdot w_3 \cdots \quad \text{where } p = (\lambda_b\,(\mathbf{f_0}\,b){=}\textbf{N-aD}, (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\mathbf{N}, (\mathbf{f_0}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\mathbf{T}, (\mathbf{f_0}{\circ}\mathbf{f_A}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\mathbf{T})$$

This function can be drawn as a path from an expected syntactic node to the root, skipping sequences of left- and right-children:



This may also be thought of as a sequence of nested **incomplete sign** fragments **T/T** and **N/N-aD**.

Left-corner parsers rarely need more than **four** incomplete sign fragments (Schuler et al., 2010).

4.4 Typed store functions as a shorthand for prediction hierarchies

Cued association structures for prediction hierarchies take up a lot of space:

$p \cdot w_1 \cdot w_2 \cdot w_3 \cdots$ where $p = (\lambda_b\,(\mathbf{f_0}\,b){=}\mathbf{N\text{-}aD}, (\mathbf{f_0}{\circ}\mathbf{f_A}\,b){=}\mathbf{N}, (\mathbf{f_0}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\mathbf{T}, (\mathbf{f_0}{\circ}\mathbf{f_A}{\circ}\mathbf{f_B}{\circ}\mathbf{f_A}\,b){=}\mathbf{T})$

Let's shorten the syntactic specifications of prediction hierarchies using typed **store functions** g:

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

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

We base our definitions on terms from typed lambda calculus (Church, 1940):

- Typed lambda calculus is defined over **base objects** called 'individuals' or 'entities'.
- Base objects of semantic expressions from the last lecture notes are **things** (stimuli sources).
- Base objects of store functions defined here are **signified referential states**.
- **Properties** p are functions from entities (e.g. signified ref. states) to truth values.

We now define the following:

- **Type constants** $\mathbf{x}_\alpha$, $\mathbf{x}_\beta$, ... in cued assoc. structures (properties over signs a, b, ...) are entities.
- **Properties** over signified ref. states x, y, ... have subtypes α, β, ..., specifying syntactic type (types provide syntactic specifications, property functions provide semantic specifications).
- **Holes** h are functions from properties to properties (e.g. from **N** to **T**, above).
- **Store functions** g are functions from properties and holes to properties.

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:

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

Derivation of translation:

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

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

4.5 Left-corner parsing operations (Johnson-Laird, 1983)

Left-corner parsing uses four kinds of operations:

1. first, two possible outcomes of a **fork** decision – whether to use a **word** to **expand** the store:
   - yes-fork – use the next word to increase the size of the store

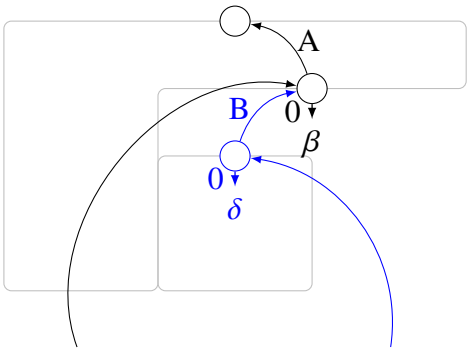
$$\frac{g:\beta\to\Gamma\ \cdot\ w:\omega\ \cdots}{\lambda_{h:\delta\to\beta}\,(g\,(h\,(\lambda_x\,\mathbf{true}))):\Gamma\ \cdots}\ \overbrace{\beta\underset{G}{\overset{+}{\to}}\delta\ldots,}^{\text{left corner}}\ \overbrace{\delta\underset{G}{\to}\omega}^{\text{lexical rule}} \qquad \text{(+F)}$$

- no-fork – use the next word to match the lowest store element

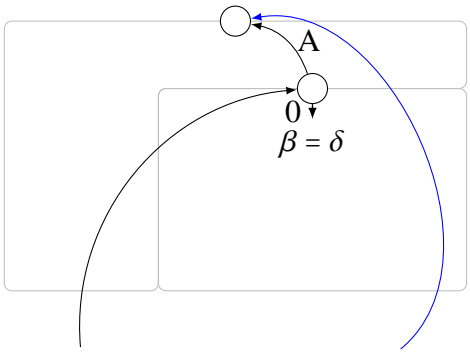
$$\frac{g:\beta\to\Gamma\ \cdot\ w:\omega\ \cdots}{(g\,(\lambda_x\,\mathbf{true})):\Gamma\ \cdots}\ \overbrace{\beta=\delta,}^{\text{match}}\ \overbrace{\delta\underset{G}{\to}\omega}^{\text{lexical rule}} \qquad (-\text{F})$$

graphically:

+F:



−F:



2. then, two possible outcomes of a **join** decision – whether to use a **rule** to **reduce** the store:

- yes-join – use a grammatical inference rule to decrease the size of the store

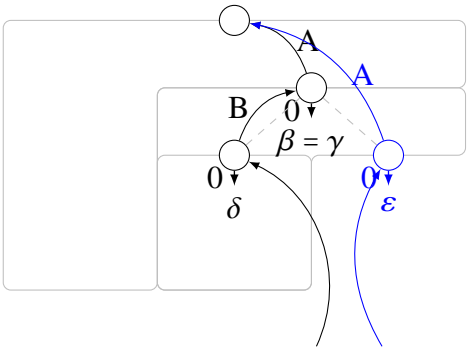
$$\frac{g:(\delta\to\beta)\to\Gamma\ \cdots}{\lambda_{q:\epsilon}\,(g\,(\lambda_{p:\delta}\,\lambda_x\,\mathbf{true})):\Gamma\ \cdots}\ \overbrace{\beta=\gamma,}^{\text{match}}\ \overbrace{\gamma\underset{G}{\to}\delta\ \varepsilon}^{\text{grammar rule}} \qquad (+\text{J})$$

- no-join – use a grammatical inference rule to extend the lowest store element

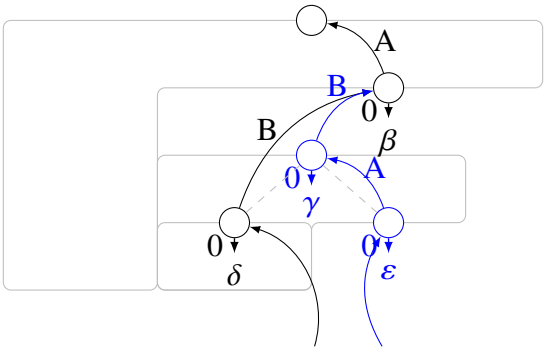
$$\frac{g:(\delta\to\beta)\to\Gamma\ \cdots}{\lambda_{q:\epsilon}\,\lambda_{h:\gamma\to\beta}\,(g\,(\lambda_{p:\delta}\,h\,(\lambda_x\,\mathbf{true}))):\Gamma\ \cdots}\ \overbrace{\beta\overset{+}{\underset{G}{\to}}\gamma\ldots,}^{\text{left corner}}\ \overbrace{\gamma\underset{G}{\to}\delta\ \varepsilon}^{\text{grammar rule}} \qquad (-\text{J})$$

graphically:

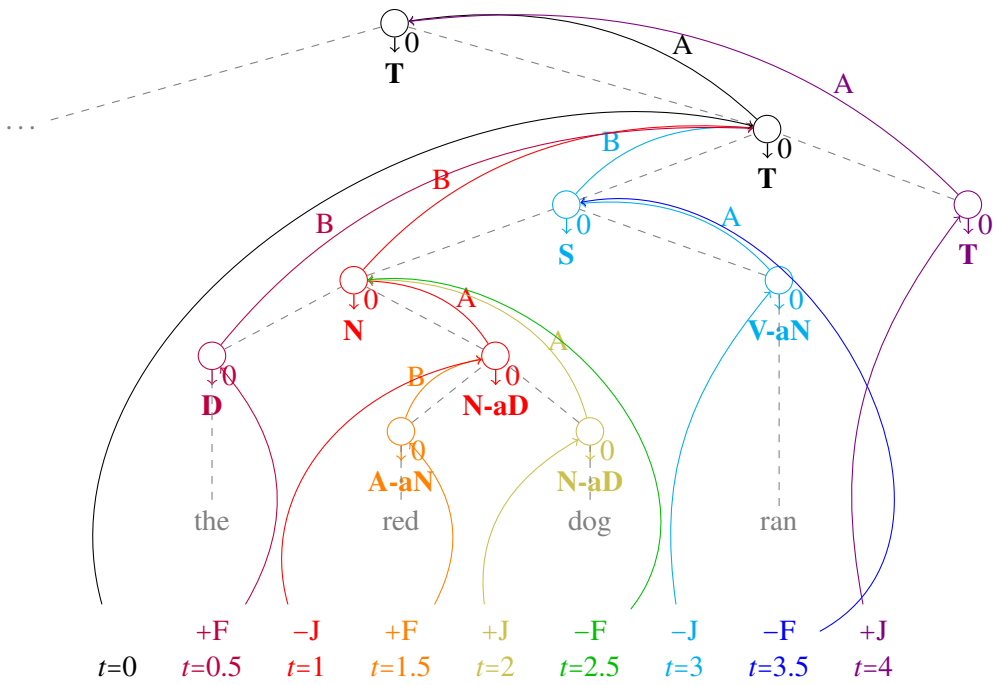
+J:



−J:



Sample run – notice process is monotonic, nothing gets deleted, just dereferenced:



We now generalize these operations to use (learned) inference rules r with semantic constraints:

1. **fork** decision, using **lexical** inference rule:

   - yes-fork ('shift without match' in Johnson-Laird terms):

$$\frac{g:\beta\to\Gamma\ \cdot\ w:\omega\ \cdots}{(r\,g\,w):(\delta\to\Delta')\to\Gamma'\ \cdots}\ r:\overbrace{(\beta\to\Gamma)}^{\text{old store}}\to\overbrace{\omega}^{\text{word}}\to\overbrace{(\delta\to\Delta')\to\Gamma'}^{\text{new store}}\ \in\ R, \qquad (+\text{F})$$

   - no-fork ('shift with match' in Johnson-Laird terms – $(\lambda_p\,p)$ is the match):

$$\frac{g:\beta\to\Gamma\ \cdot\ w:\omega\ \cdots}{(r\,g\,w\,(\lambda_p\,p)):\Gamma'\ \cdots}\ r:\overbrace{(\beta\to\Gamma)}^{\text{old store}}\to\overbrace{\omega}^{\text{word}}\to\overbrace{(\beta\to\beta)\to\Gamma'}^{\text{new store}}\ \in\ R, \qquad (-\text{F})$$

2. **join** decision, using **grammatical** inference rule:

   - yes-join ('predict with match' in Johnson-Laird terms – $(\lambda_p\,p)$ is the match):

$$\frac{g:(\delta\to\Delta)\to\Gamma\ \cdots}{\lambda_{q_{1\ldots n}:\varepsilon_{1\ldots n}}(r\,g\,q_{1\ldots n}\,(\lambda_p\,p)):\Gamma'\ \cdots}\ r:\overbrace{((\delta\to\Delta)\to\Gamma)}^{\text{old store}}\to\overbrace{\varepsilon_{1\ldots n}\to(\gamma\to\gamma)\to\Gamma'}^{\text{new store}}\ \in\ R, \qquad (+\text{J})$$

   - no-join ('predict without match' in Johnson-Laird terms):

$$\frac{g:(\delta\to\Delta)\to\Gamma\ \cdots}{(r\,g):\varepsilon_{1\ldots n}\to(\gamma\to\Delta')\to\Gamma'\ \cdots}\ r:\overbrace{((\delta\to\Delta)\to\Gamma)}^{\text{old store}}\to\overbrace{\varepsilon_{1\ldots n}\to(\gamma\to\Delta')\to\Gamma'}^{\text{new store}}\ \in\ R, \qquad (-\text{J})$$

4.6 Lexical inference rules (Nguyen et al., 2012)

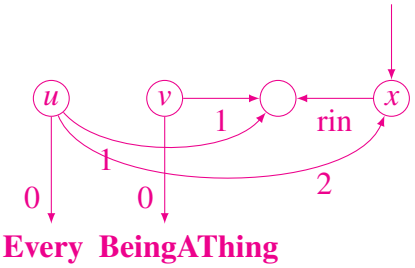
Lexical inference rules integrate word meanings into a cued association structure.

They are of type $\overbrace{(\beta \to \Gamma)}^{\text{old store}} \to \overbrace{\omega}^{\text{word}} \to \overbrace{((\overbrace{\delta}^{\text{preterminal}} \to \Delta') \to \Gamma')}^{\text{new store}}$.

For example, the word '*everything*' would use following lexical inference rule:

$$\lambda_{g:\beta\to\Gamma}\ \lambda_{w:\mathbf{everything}}\ \lambda_{h:\mathbf{N}\to\beta}\ (g{\circ}h\,(\lambda_x\,\exists_u\,(\mathbf{f_0}\,u){=}\mathbf{Every},(\mathbf{f_1}\,u){=}(\mathbf{f_{rin}}\,x),(\mathbf{f_2}\,u){=}x,\ \exists_v\,(\mathbf{f_0}\,v){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v){=}(\mathbf{f_{rin}}\,x)))\!:\!\Gamma\ \in\ R$$

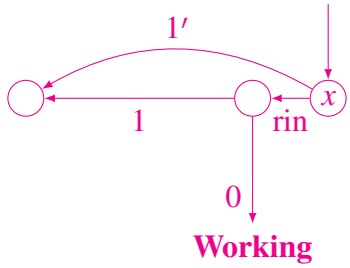
graphically (top arrow indicates lambda input / attachment site):



and the word '*works*' would use the rule:

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

graphically (top arrow indicates lambda input / attachment site):



Here, predications like **Working** are in restrictors. This is because they can be quantified.

For example, using '*usually*' as a (raised) quantifier:

(9) a. *I usually like rainstorms.* (means *I like most rainstorms.*)
b. *I usually like that it rains / for it to rain.* (means *I like most instances of it raining.*)

4.7 Grammatical inference rules (Nguyen et al., 2012)

Grammatical inference rules integrate compositional meaning into a cued association structure:

1. 'Binary' inference rules build signs of type γ from left and right children of type δ and ε.

   These rules are of type $((\overbrace{\delta}^{\text{l. child}} \to \Delta) \to \Gamma) \to (\overbrace{\varepsilon_1 \to \ldots \to \varepsilon_n \to (\overbrace{\gamma}^{\text{parent}} \to \Delta')}^{\text{r. child}} \to \Gamma')$.

2. 'Unary' inference rules build signs of type δ' from single chidren of type δ.

   These rules are of type $((\overbrace{\delta}^{\text{child}} \to \Delta) \to \Gamma) \to ((\overbrace{\delta'}^{\text{parent}} \to \Delta') \to \Gamma')$.

Lexical inference rules may combine with one or more unary rules:

$$\text{for } \overbrace{r : (\beta \to \Gamma) \to \omega \to (\delta \to \Delta) \to \Gamma'}^{\text{lexical inference rule}}, \ \overbrace{r' : ((\delta \to \Delta) \to \Gamma') \to (\delta' \to \Delta') \to \Gamma''}^{\text{unary rule}} \in R,$$
$$\lambda_{g:\beta\to\Gamma}\, r' \circ (r\, g) : \omega \to (\delta' \to \Delta') \to \gamma'' \ \in \ R \tag{1}$$

Grammatical inference rules may also combine with one or more unary rules:

$$\text{for } \overbrace{r : ((\delta \to \Delta) \to \Gamma) \to \varepsilon_1 \to \ldots \to \varepsilon_n \to (\gamma \to \Delta') \to \Gamma'}^{\text{grammatical inference rule}}, \ \overbrace{r' : ((\gamma \to \Delta') \to \Gamma') \to (\gamma' \to \Delta'') \to \Gamma''}^{\text{unary rule}} \in R,$$
$$\lambda_{g:(\delta\to\Delta)\to\Gamma}\, r' \circ (r\, g) : \varepsilon_1 \to \ldots \to \varepsilon_n \to (\gamma' \to \Delta'') \to \Gamma'' \ \in \ R \tag{2}$$

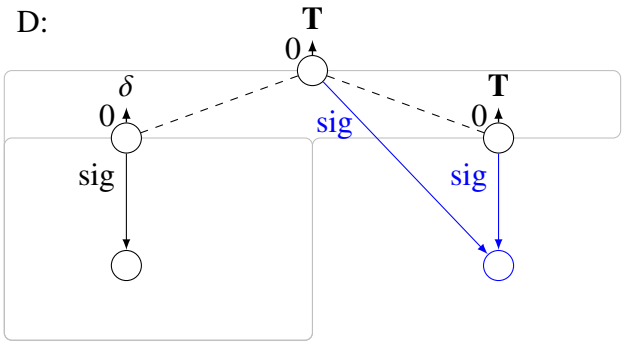
Below is a set of grammatical inference rules for English ...

4.7.1 Discourse attachment (binary)

First, we introduce a naive discourse attachment rule to apply at the end of each sentence:

$$\lambda_{g:(\delta\to\mathbf{T})\to\mathbf{T}}\ \lambda_{q:\mathbf{T}}\ \lambda_{h:\mathbf{T}\to\mathbf{T}}\ (g\,(\lambda_p\, h\,(\lambda_y\, \exists_x\,(p\,x), (q\,y))))) : \Gamma \ \in \ R \tag{D}$$

Graphically:



This doesn't connect anything in the sentence to anything outside the sentence.

(Other discourse connectives are possible, and desirable.)

4.7.2 Argument attachment (binary)

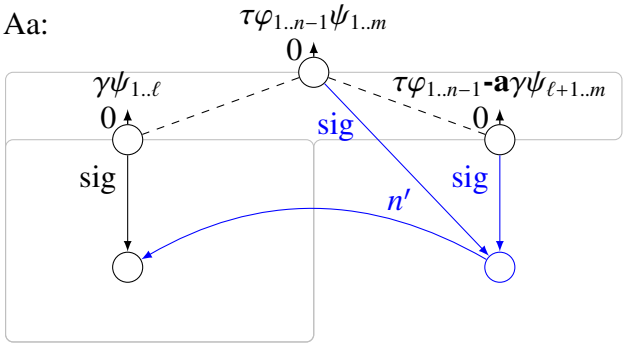
Add rules to attach one sign as the nth argument of another using a cued association labeled n':

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

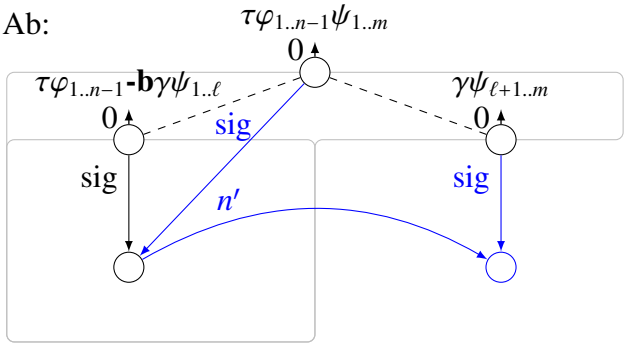
$$\lambda_{g:(\tau\varphi_{1..n-1}\textbf{-b}\gamma\psi_{1..\ell}\to\Delta)\to\Gamma}\ \lambda_{q:\gamma\psi_{\ell+1..m}}\ \lambda_{h:\tau\varphi_{1..n-1}\psi_{1..m}\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),(\mathbf{f}_{n'}\,x){=}y)))\!:\!\Gamma\ \in\ R \qquad \text{(Ab)}$$

Graphically:

Aa:



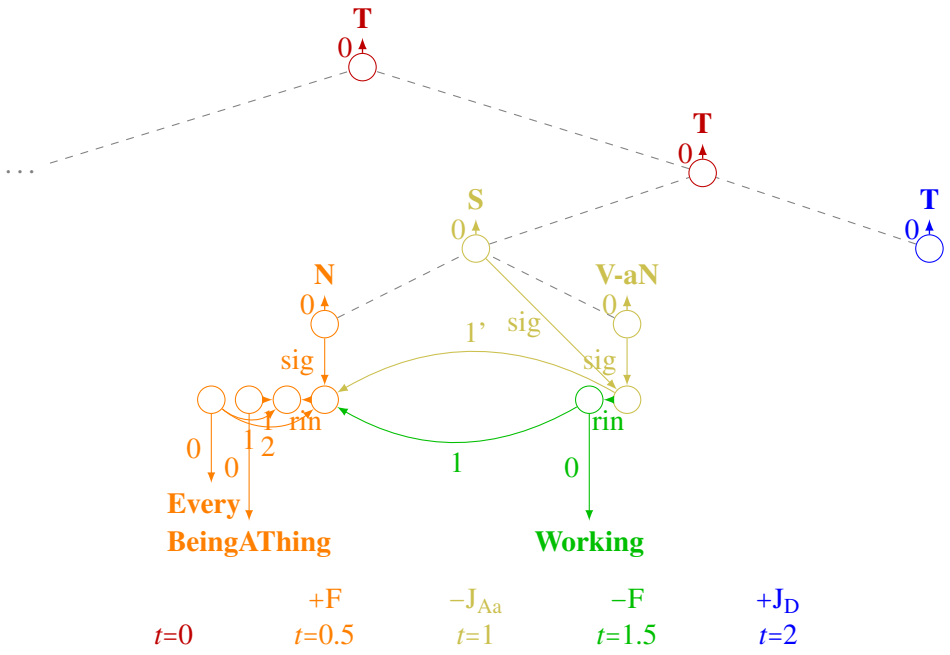
Ab:



For example, a parse of the sentence '*Everything works*:'

$$\dfrac{\dfrac{\dfrac{\dfrac{\lambda_{q,x_0}\,(q\,x_0)\!:\!\mathbf{T}\to\mathbf{T}\qquad \textbf{unit}\!:\!\textbf{everything}}{\lambda_{h,x_0}\,h\,(\lambda_{x_1}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\textbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\textbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1))\!:\!(\mathbf{N}\to\mathbf{T})\to\mathbf{T}}\ {+}\text{F}}{\lambda_{q,h,x_0}\,h\,(\lambda_{x_2}\exists_{x_1}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\textbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\textbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (q\,x_2),(\mathbf{f_{1'}}\,x_2){=}x_1)\!:\!\textbf{V-aN}\to(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\ {-}\text{J}_{\text{Aa}}}\qquad \textbf{unit}\!:\!\textbf{works}}{\lambda_{h,x_0}\,h\,(\lambda_{x_2}\exists_{x_1}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\textbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\textbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\textbf{Working},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}x_1,(\mathbf{f_{1'}}\,x_2){=}x_1)\!:\!(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\ {-}\text{F}}{\lambda_{q,x_0}\,\exists_{x_1,x_2}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\textbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\textbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\textbf{Working},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}x_1,(\mathbf{f_{1'}}\,x_2){=}x_1,(q\,x_0))\!:\!\mathbf{T}\to\mathbf{T}}\ {+}\text{J}_{\text{D}}$$

Graphical representation of resulting cued association structure:



4.7.3 Auxiliary attachment (binary)

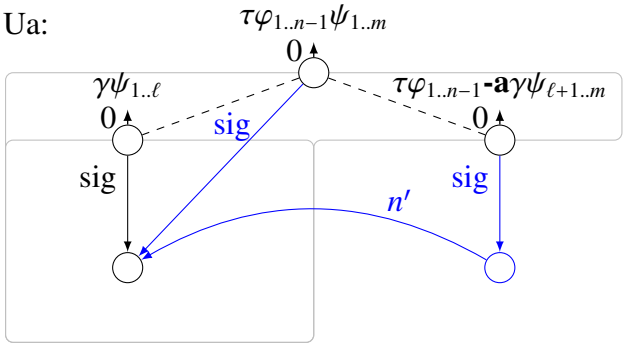
This model also allows attachment of arguments that keeps the argument in attentional focus:

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

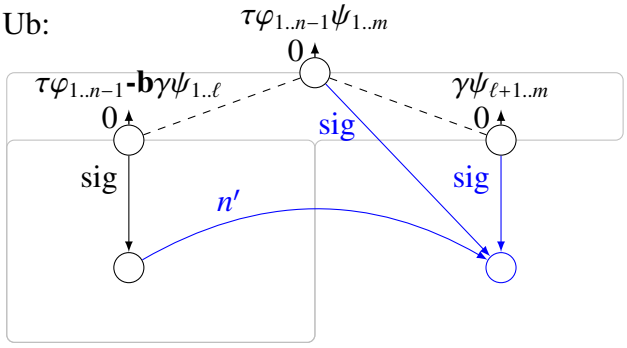
$$\lambda_{g:(\tau\varphi_{1..n-1}\textbf{-b}\gamma\psi_{1..\ell}\to\Delta)\to\Gamma}\ \lambda_{q:\gamma\psi_{\ell+1..m}}\ \lambda_{h:\tau\varphi_{1..n-1}\psi_{1..m}\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(q\,y),(\mathbf{f}_{n'}\,x){=}y)))\!:\!\Gamma\ \in\ R \qquad \text{(Ub)}$$

Graphically:

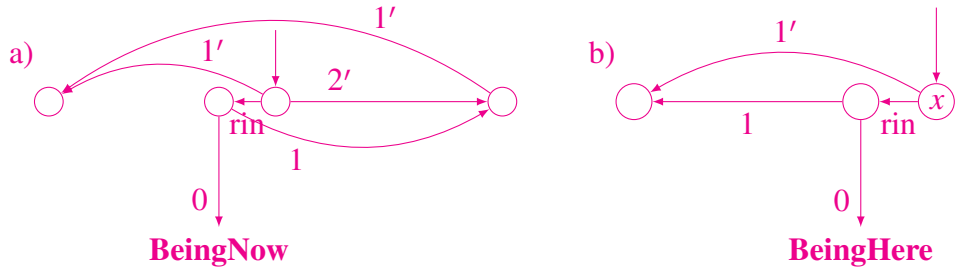
Ua:



Ub:



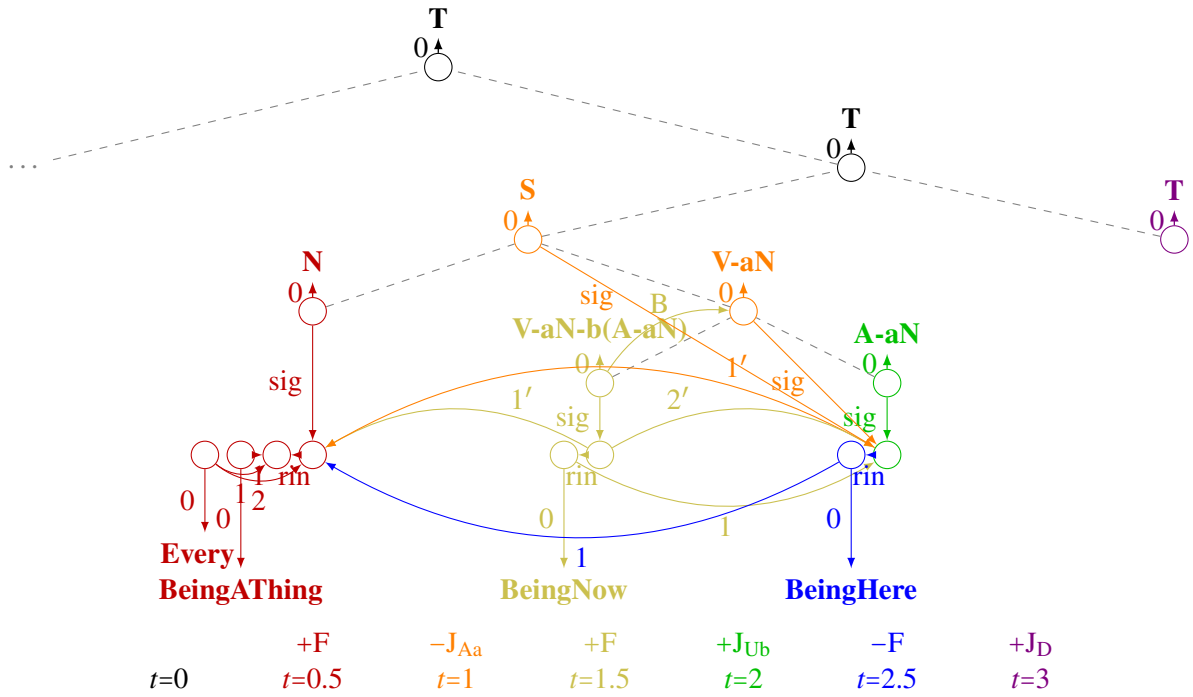
For example, with the following lexical inference rules for '*is*' and '*here*':



For example, a parse of the sentence '*Everything is here,*':

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\lambda_{q,x_0}\,(q\,x_0):\mathbf{T}\to\mathbf{T} \qquad \mathbf{unit}:\mathbf{everything}}{\lambda_{h,x_0}\,h\,(\lambda_{x_1}\exists_{u_1}\,(\mathbf{f_0}\,u_1)=\mathbf{Every},(\mathbf{f_1}\,u_1)=(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1)=x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1)=\mathbf{BeingAThing},(\mathbf{f_1}\,v_1)=(\mathbf{f_{rin}}\,x_1)):(\mathbf{N}\to\mathbf{T})\to\mathbf{T}}\,{+}\mathrm{F}}{\lambda_{q,h,x_0}\,h\,(\lambda_{x_2}\,\exists_{x_1}\exists_{u_1}\,(\mathbf{f_0}\,u_1)=\mathbf{Every},(\mathbf{f_1}\,u_1)=(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1)=x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1)=\mathbf{BeingAThing},(\mathbf{f_1}\,v_1)=(\mathbf{f_{rin}}\,x_1),\ (q\,x_2),(\mathbf{f_{1'}}\,x_2)=x_1):\mathbf{V\text{-}aN}\to(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{-}\mathrm{J_{Aa}} \qquad \mathbf{unit}:\mathbf{is}}{\lambda_{h,x_0}\,h\,(\lambda_{x_2}\,\exists_{x_1}\exists_{u_1}\,(\mathbf{f_0}\,u_1)=\mathbf{Every},(\mathbf{f_1}\,u_1)=(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1)=x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1)=\mathbf{BeingAThing},(\mathbf{f_1}\,v_1)=(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2))=\mathbf{BeingNow},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2))=(\mathbf{f_{2'}}\,x_2),\ (\mathbf{f_{1'}}\,x_2)=x_1,(\mathbf{f_{1'}}\,(\mathbf{f_{2'}}\,x_2))=x_1):(\mathbf{V\text{-}aN\text{-}b(A\text{-}aN)}\to\mathbf{V\text{-}aN})\to(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{+}\mathrm{F}}{\lambda_{q,h,x_0}\,h\,(\lambda_{x_3}\,\exists_{x_1,x_2}\exists_{u_1}\,(\mathbf{f_0}\,u_1)=\mathbf{Every},(\mathbf{f_1}\,u_1)=(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1)=x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1)=\mathbf{BeingAThing},(\mathbf{f_1}\,v_1)=(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2))=\mathbf{BeingNow},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2))=(\mathbf{f_{2'}}\,x_2)=x_3,\ (\mathbf{f_{1'}}\,x_2)=x_1,(\mathbf{f_{1'}}\,(\mathbf{f_{2'}}\,x_2))=x_1,(q\,x_3)):\mathbf{A\text{-}aN}\to(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{+}\mathrm{J_{Ub}} \qquad \mathbf{unit}:\mathbf{here}}{\lambda_{h,x_0}\,h\,(\lambda_{x_3}\,\exists_{x_1,x_2}\exists_{u_1}\,(\mathbf{f_0}\,u_1)=\mathbf{Every},(\mathbf{f_1}\,u_1)=(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1)=x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1)=\mathbf{BeingAThing},(\mathbf{f_1}\,v_1)=(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2))=\mathbf{BeingNow},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2))=(\mathbf{f_{2'}}\,x_2)=x_3,\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_3))=\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_3))=x_1,\ (\mathbf{f_{1'}}\,x_2)=x_1,(\mathbf{f_{1'}}\,(\mathbf{f_{2'}}\,x_2))=x_1):(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{-}\mathrm{F}}{\lambda_{q,x_0}\,\exists_{x_1,x_2,x_3}\exists_{u_1}\,(\mathbf{f_0}\,u_1)=\mathbf{Every},(\mathbf{f_1}\,u_1)=(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1)=x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1)=\mathbf{BeingAThing},(\mathbf{f_1}\,v_1)=(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2))=\mathbf{BeingNow},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2))=(\mathbf{f_{2'}}\,x_2)=x_3,\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_3))=\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_3))=x_1,\ (\mathbf{f_{1'}}\,x_2)=x_1,(\mathbf{f_{1'}}\,(\mathbf{f_{2'}}\,x_2))=x_1,(q\,x_0)):\mathbf{T}\to\mathbf{T}}\,{+}\mathrm{J_D}$$

Graphical representation of resulting cued association structure:



(This rule is also used for quantifiers.)

4.7.4 Modifier attachment (binary)

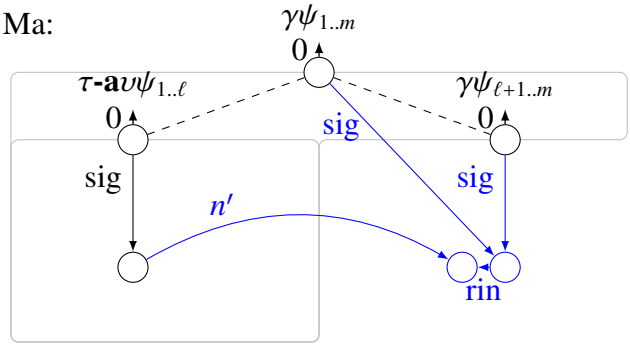
This model also allows attachment of modifiers:

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

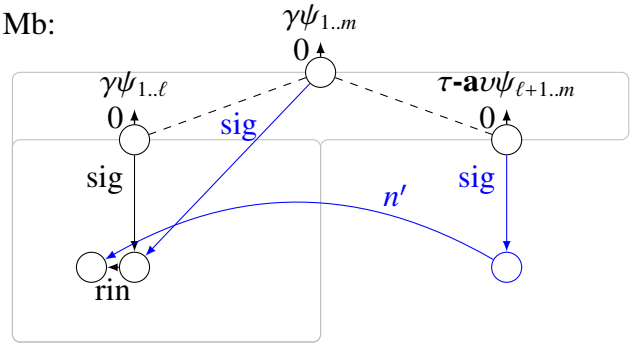
$$\lambda_{g:(\gamma\psi_{1..\ell}\to\Delta)\to\Gamma}\ \lambda_{q:\tau\text{-}\mathbf{a}\upsilon\psi_{\ell+1..m}}\ \lambda_{h:\gamma\psi_{1..m}\to\Delta}$$
$$(g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),(\mathbf{f}_{\mathbf{rin}}\,x)=(\mathbf{f}_{\mathbf{1}'}\,y))))\!:\!\Gamma\ \in\ R \qquad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

Graphically:

Ma:



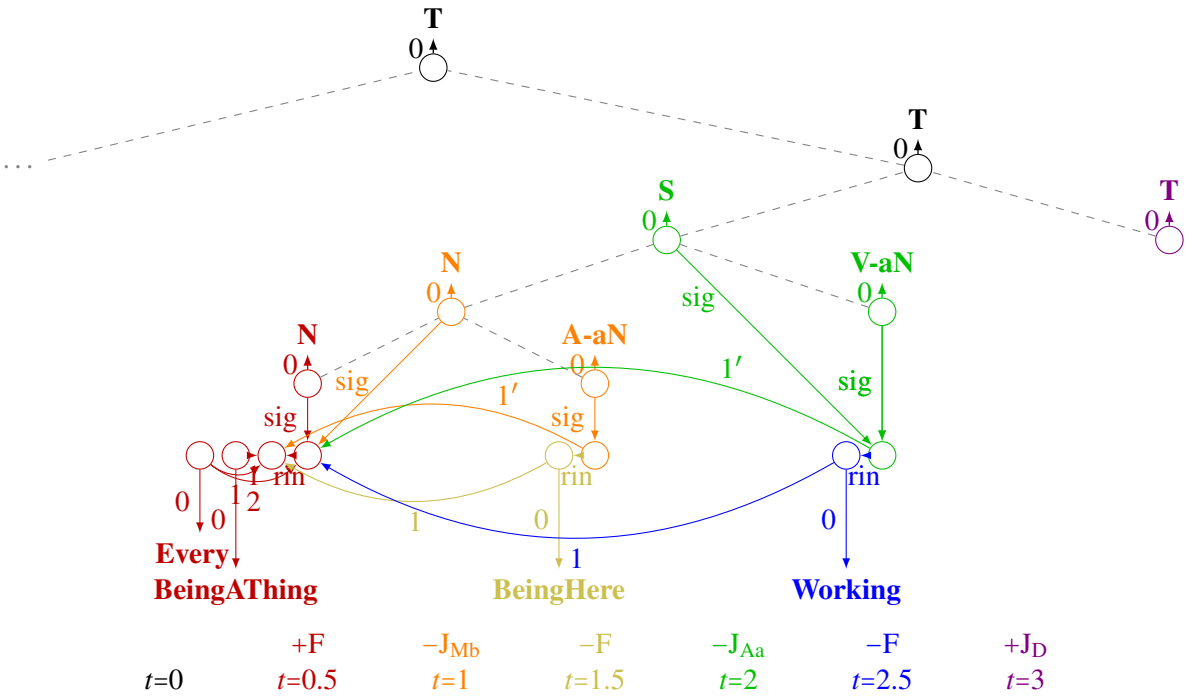
Mb:



For example, a parse of the sentence '*Everything here works*':

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\lambda_{q,x_0}\,(q\,x_0):\mathbf{T}\to\mathbf{T} \qquad \mathbf{unit}:\mathbf{everything}}{\lambda_{h,x_0}\,h\,(\lambda_{x_1}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1)):(\mathbf{N}\to\mathbf{T})\to\mathbf{T}}\,{+}\mathrm{F}}{\lambda_{q,h,x_0}\,h\,(\lambda_{x_2}\exists_{x_1}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (q\,x_2),(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1)):\mathbf{A\text{-}aN}\to(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{-}\mathrm{J_{Mb}} \qquad \mathbf{unit}:\mathbf{here}}{\lambda_{h,x_0}\,h\,(\lambda_{x_1}\exists_{x_2}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1)):(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{-}\mathrm{F}}{\lambda_{q,h,x_0}\,h\,(\lambda_{x_3}\exists_{x_1,x_2}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1),\ (q\,x_3),(\mathbf{f_{1'}}\,x_3){=}x_1):\mathbf{V\text{-}aN}\to(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{-}\mathrm{J_{Aa}} \qquad \mathbf{unit}:\mathbf{works}}{\lambda_{h,x_0}\,h\,(\lambda_{x_3}\exists_{x_1,x_2}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_3)){=}\mathbf{Working},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_3)){=}(\mathbf{f_{1'}}\,x_3){=}x_1):(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{-}\mathrm{F}}{\lambda_{q,x_0}\,\exists_{x_1,x_2,x_3}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_3)){=}\mathbf{Working},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_3)){=}(\mathbf{f_{1'}}\,x_3){=}x_1,(q\,x_0)):\mathbf{T}\to\mathbf{T}}\,{+}\mathrm{J_D}$$

Graphical representation of resulting cued association structure:



This special set of modifier rules allows post-copular predicatives:

(10) a. A gear failure was [$_{\mathbf{A\text{-}aN}}$ similar to this].
b. A gear failure was [$_{\mathbf{A\text{-}aN}}$ in the pump].
c. A gear failure was [$_{\mathbf{A\text{-}aN}}$ slowing the pump].

to also be used as post-nominal modifiers:

(11) a. A gear failure [$_{\textbf{A-aN}}$ similar to this] was reported.
b. A gear failure [$_{\textbf{A-aN}}$ in the pump] was reported.
c. A gear failure [$_{\textbf{A-aN}}$ slowing the pump] was reported.

and, in a similar context, as post-verbal modifiers:

(12) a. A gear failed [$_{\textbf{R-aN}}$ similarly to this].
b. A gear failed [$_{\textbf{R-aN}}$ in the pump].
c. A gear failed [$_{\textbf{R-aN}}$ slowing the pump].

4.7.5 Conjunct attachment (binary)

This model also allows attachment of left, middle, and right conjuncts:

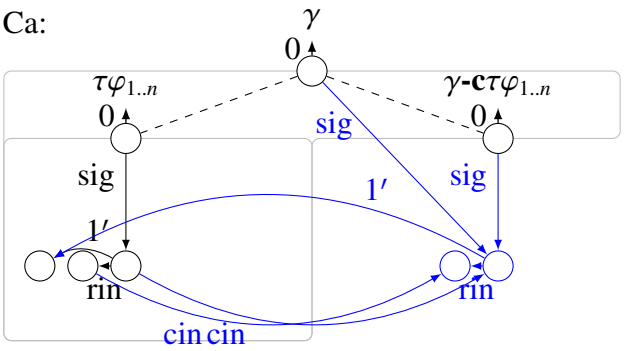
$$\lambda_{g:(\tau\varphi_{1..n}\to\Delta)\to\Gamma}\ \lambda_{q:\gamma\text{-}\mathbf{c}\tau\varphi_{1..n}}\ \lambda_{h:\gamma\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(q\,y),\ (\mathbf{f}_{\mathbf{1}'}\,x)=(\mathbf{f}_{\mathbf{1}'}\,y),\ldots,(\mathbf{f}_{n'}\,x)=(\mathbf{f}_{n'}\,y),(\mathbf{f}_{\mathbf{cin}}\,x)=y,(\mathbf{f}_{\mathbf{cin}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

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

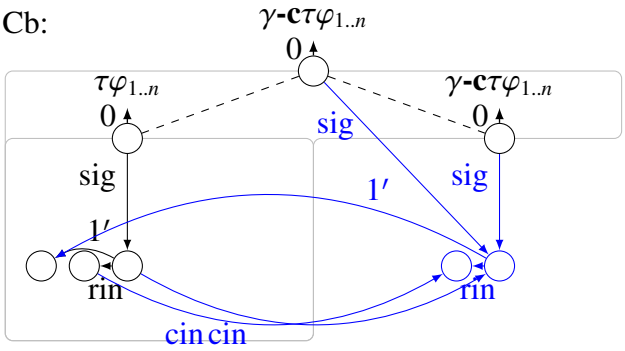
$$\lambda_{g:(\gamma\text{-}\mathbf{d}\tau\varphi_{1..n}\to\Delta)\to\Gamma}\ \lambda_{q:\tau\varphi_{1..n}}\ \lambda_{h:\gamma\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),\ (\mathbf{f}_{\mathbf{1}'}\,x)=(\mathbf{f}_{\mathbf{1}'}\,y),\ldots,(\mathbf{f}_{n'}\,x)=(\mathbf{f}_{n'}\,y),x=(\mathbf{f}_{\mathbf{cin}}\,y),(\mathbf{f}_{\mathbf{rin}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

Graphically:

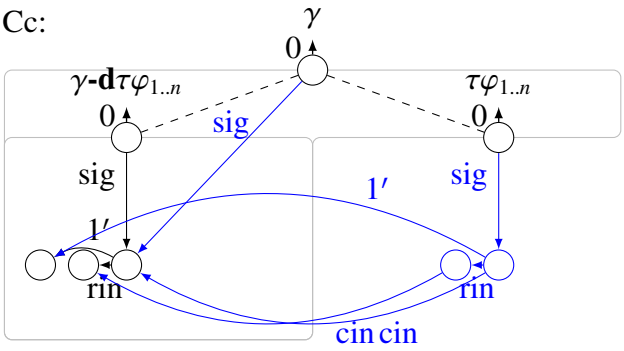
Ca:



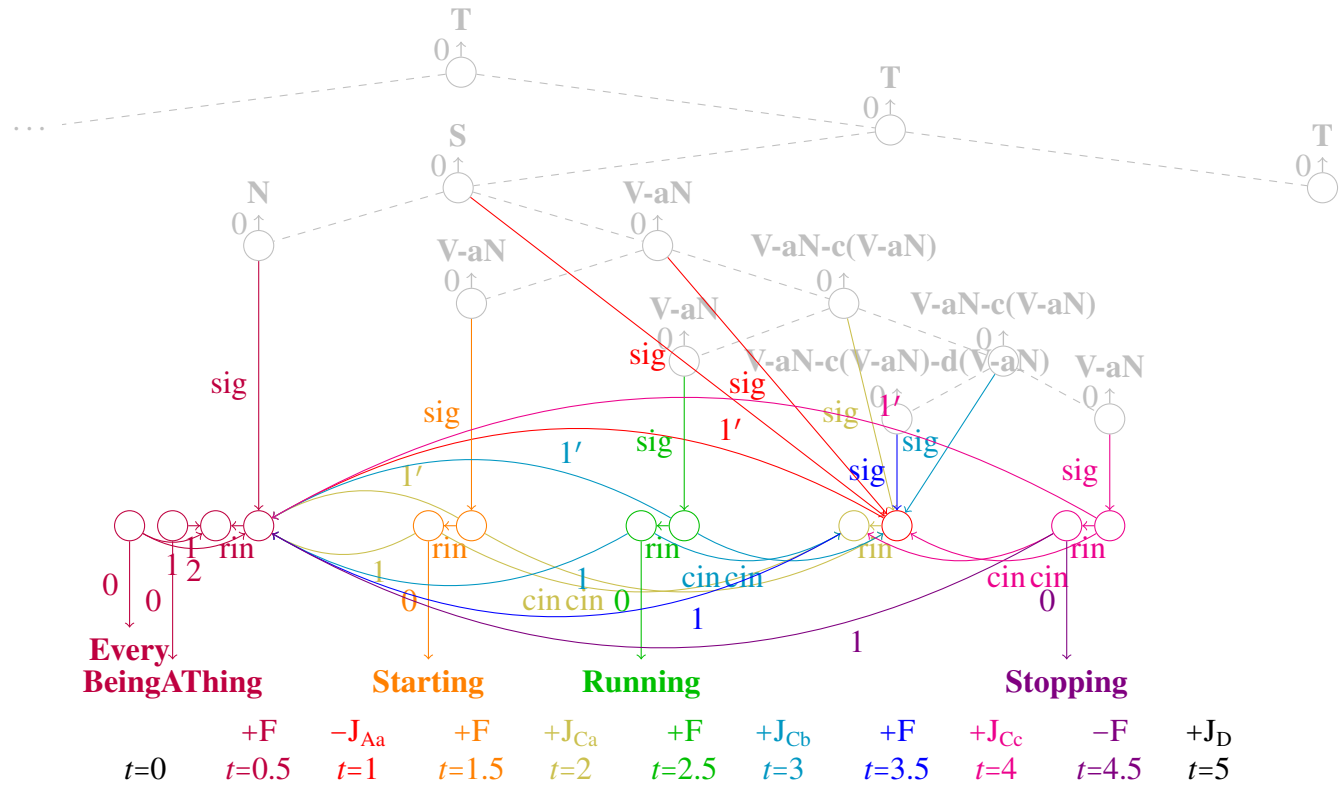
Cb:



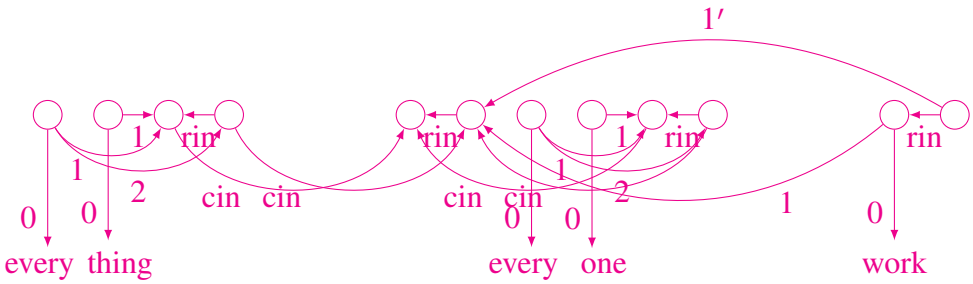
Cc:



Example: '*Everything starts, runs and stops.*'



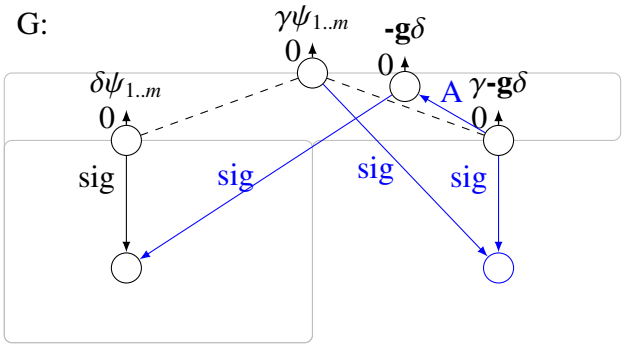
Example: '*Everything and everyone works.*'



4.7.6 Gap-filler attachment (binary)

This model also allows attachment of gap fillers, creating non-local dependencies:

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



4.7.7 Non-local dependency removal (unary)

Non-local dependencies can be re-used, so we will need a separate rule to clean them up (learned to apply only if dependency no longer occurs in store: $\psi \notin \alpha, \beta, \alpha', \beta', \ldots$).

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

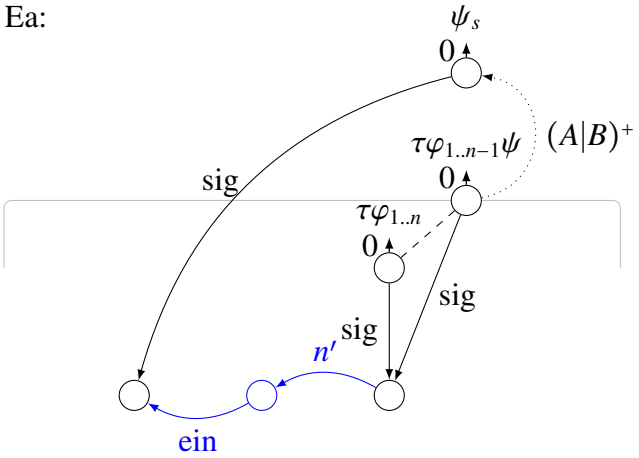
4.7.8 Extraction (unary)

Non-local dependencies can then be used as arguments or modifiers:

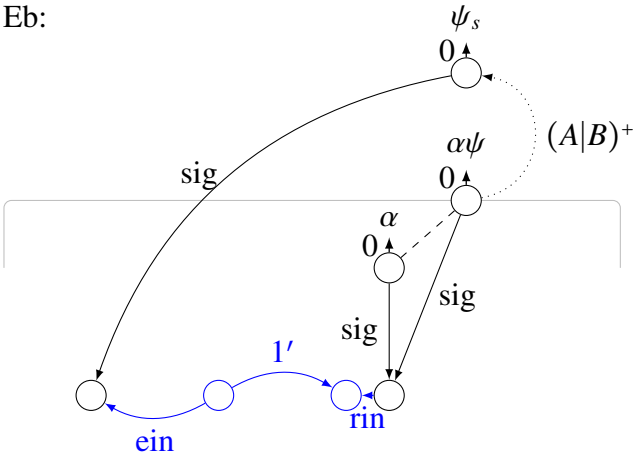
$$\lambda_{g:(\tau\varphi_{1..n}\to\Delta)\to\ldots\to\psi\to\Gamma}\ \lambda_{h:\tau\varphi_{1..n-1}\psi\to\Delta}\ \ldots\ \lambda_{q':\psi}$$
$$\exists_y\,(g\,(\lambda_p\,h\,(\lambda_x\,(p\,x),(\mathbf{f}_{n'}\,x){=}y))\ldots(\lambda_z\,(q'\,z),(\mathbf{f_{ein}}\,y){=}z)):\Gamma\ \in\ R \qquad \text{(Ea)}$$

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

Ea:



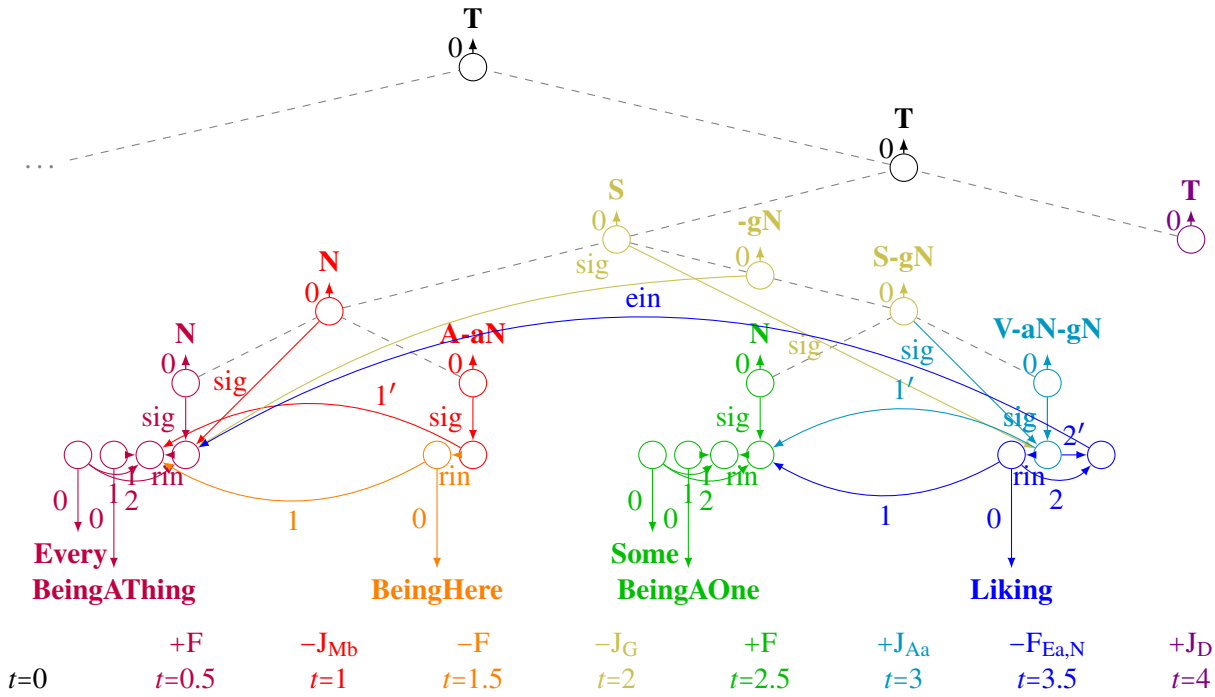
Eb:



For example:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\lambda_{q,x_0}\,(q\,x_0):\mathbf{T}\to\mathbf{T}\qquad \mathbf{unit}:\mathbf{everything}}{\lambda_{h,x_0}\,h\,(\lambda_{x_1}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1)):(\mathbf{N}\to\mathbf{T})\to\mathbf{T}}\,{+}\mathrm{F}}{\lambda_{q,h,x_0}\,h\,(\lambda_{x_2}\exists_{x_1}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (q\,x_2),(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1)):\mathbf{A\text{-}aN}\to(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{-}\mathrm{J_{Mb}}}{\lambda_{h,x_0}\,h\,(\lambda_{x_1}\exists_{x_2}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1)):(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{-}\mathrm{F}}{\lambda_{q,h,x_0}\,h\,(\lambda_{x_3}\exists_{x_1,x_2}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1),\ (q\,x_3)):\mathbf{S\text{-}gN}\to\mathbf{\text{-}gN}\to(\mathbf{S}\to\mathbf{T})\to\mathbf{T}\qquad \mathbf{unit}:\mathbf{someone}}\,{+}\mathrm{F}}{\lambda_{h,x_0}\,h\,(\lambda_{x_3}\exists_{x_1,x_2}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1),\ \exists_{u_3}\,(\mathbf{f_0}\,u_3){=}\mathbf{Some},(\mathbf{f_1}\,u_3){=}(\mathbf{f_{rin}}\,x_3),(\mathbf{f_2}\,u_1){=}x_3,\ \exists_{v_3}\,(\mathbf{f_0}\,v_3){=}\mathbf{BeingAOne},(\mathbf{f_1}\,v_3){=}(\mathbf{f_{rin}}\,x_3)):(\mathbf{N}\to\mathbf{S}-\mathbf{gN})\to\mathbf{\text{-}gN}\to(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{+}\mathrm{J_{Aa}}}{\lambda_{h,x_0}\,h\,(\lambda_{x_4}\exists_{x_1,x_2,x_3}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1),\ \exists_{u_3}\,(\mathbf{f_0}\,u_3){=}\mathbf{Some},(\mathbf{f_1}\,u_3){=}(\mathbf{f_{rin}}\,x_3),(\mathbf{f_2}\,u_1){=}x_3,\ \exists_{v_3}\,(\mathbf{f_0}\,v_3){=}\mathbf{BeingAOne},(\mathbf{f_1}\,v_3){=}(\mathbf{f_{rin}}\,x_3),\ (q\,x_4),(\mathbf{f_{1'}}\,x_4){=}x_3):\mathbf{V}-\mathbf{aN}-\mathbf{gN}\to\mathbf{\text{-}gN}\to(\mathbf{S}\to\mathbf{T})\to\mathbf{T}\qquad \mathbf{unit}:\mathbf{likes}}\,{-}\mathrm{F_{Ea,N}}}{\lambda_{h,x_0}\,h\,(\lambda_{x_4}\exists_{x_1,x_2,x_3}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1),\ \exists_{u_3}\,(\mathbf{f_0}\,u_3){=}\mathbf{Some},(\mathbf{f_1}\,u_3){=}(\mathbf{f_{rin}}\,x_3),(\mathbf{f_2}\,u_1){=}x_3,\ \exists_{v_3}\,(\mathbf{f_0}\,v_3){=}\mathbf{BeingAOne},(\mathbf{f_1}\,v_3){=}(\mathbf{f_{rin}}\,x_3),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_4)){=}\mathbf{Liking},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_4)){=}(\mathbf{f_{1'}}\,x_4){=}x_3,(\mathbf{f_2}\,(\mathbf{f_{rin}}\,x_4)){=}(\mathbf{f_{2'}}\,x_4),\ (\mathbf{f_{ein}}\,(\mathbf{f_{2'}}\,x_4)){=}x_1):(\mathbf{S}\to\mathbf{T})\to\mathbf{T}}\,{+}\mathrm{J_D}}{\lambda_{h,x_0}\,h\,(\lambda_{x_4}\exists_{x_1,x_2,x_3,x_4}\exists_{u_1}\,(\mathbf{f_0}\,u_1){=}\mathbf{Every},(\mathbf{f_1}\,u_1){=}(\mathbf{f_{rin}}\,x_1),(\mathbf{f_2}\,u_1){=}x_1,\ \exists_{v_1}\,(\mathbf{f_0}\,v_1){=}\mathbf{BeingAThing},(\mathbf{f_1}\,v_1){=}(\mathbf{f_{rin}}\,x_1),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_2)){=}\mathbf{BeingHere},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_2)){=}(\mathbf{f_{1'}}\,x_2){=}(\mathbf{f_{rin}}\,x_1),\ \exists_{u_3}\,(\mathbf{f_0}\,u_3){=}\mathbf{Some},(\mathbf{f_1}\,u_3){=}(\mathbf{f_{rin}}\,x_3),(\mathbf{f_2}\,u_1){=}x_3,\ \exists_{v_3}\,(\mathbf{f_0}\,v_3){=}\mathbf{BeingAOne},(\mathbf{f_1}\,v_3){=}(\mathbf{f_{rin}}\,x_3),\ (\mathbf{f_0}\,(\mathbf{f_{rin}}\,x_4)){=}\mathbf{Liking},(\mathbf{f_1}\,(\mathbf{f_{rin}}\,x_4)){=}(\mathbf{f_{1'}}\,x_4){=}x_3,(\mathbf{f_2}\,(\mathbf{f_{rin}}\,x_4)){=}(\mathbf{f_{2'}}\,x_4),\ (\mathbf{f_{ein}}\,(\mathbf{f_{2'}}\,x_4)){=}x_1):\mathbf{T}\to\mathbf{T}}$$

Graphical representation:



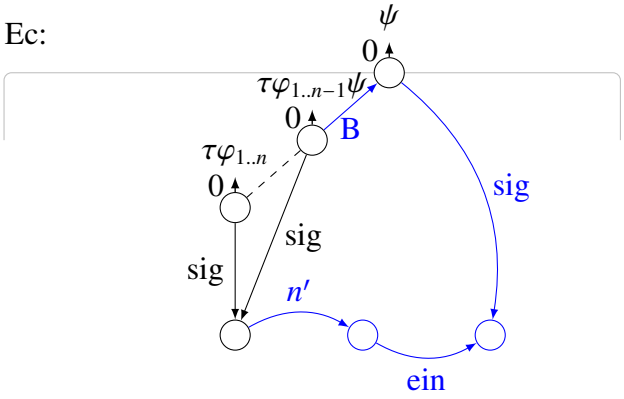
We need additional extraction rules that introduce nonlocal dependencies for arguments (Ec), modifiers (Ed), and modificands of nested nonlocal dependencies (Ee):

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

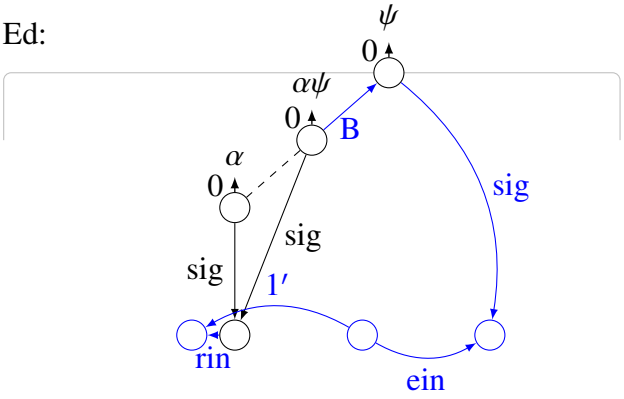
$$\lambda_{g:(\alpha\to\beta)\to\Gamma}\ \lambda_{h:\alpha\psi\to\beta}\ \lambda_{q':\psi}\ (g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q'\,(\mathbf{f}_{\mathbf{ein}}\,y)),(\mathbf{f}_{\mathbf{rin}}\,x){=}(\mathbf{f}_{\mathbf{1}'}\,y))))\!:\!\Gamma\ \in\ R \qquad \text{(Ed)}$$

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

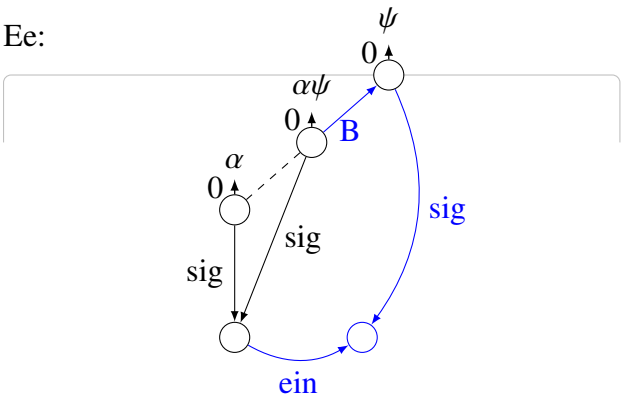
Ec:



Ed:



Ee:



4.7.9 Heavy-shift / extraposition attachment (binary)

We need to extrapose or heavy-shift arguments and modifiers, or modificands for nested nonlocal dependencies,

For example, constraints of predicates are applied to the nuclear scope sets of extraposed or heavy-shifted arguments:

(13) The spy [V-aN-hO [V-aN-hO [V-aN-bN stole] [N-hO a plan]] yesterday] [O of every new jet].

but constraints of extraposed relative clauses and adverbial comparatives are applied to the restrictor sets of modificands:

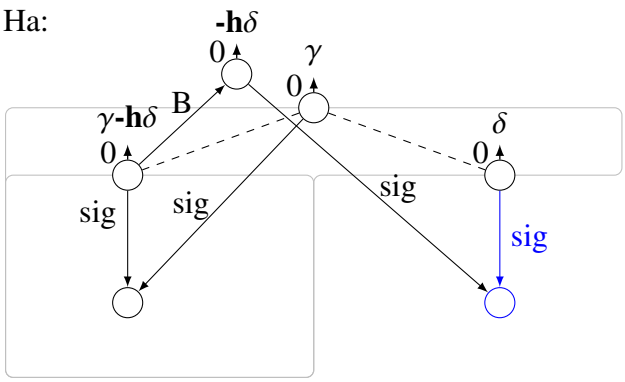
(14) a. [V-h(C-rN) [N-h(C-rN) Everything _] stops] [C-rN that _ starts].
b. [N-h(Cthan-g(A-aN)) Every [A-aN-h(Cthan-g(A-aN)) taller _] house] [Cthan-g(A-aN) than my house is wide] is gone.

Rules:

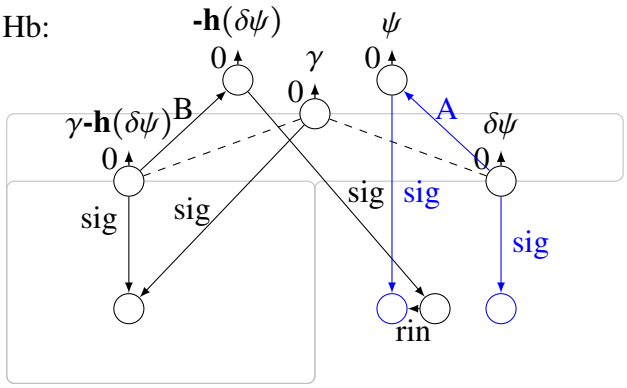
$$\lambda_{g:(\gamma\text{-}\mathbf{h}\delta\to\text{-}\mathbf{h}\delta\to\Delta)\to\Gamma}\ \lambda_{q:\varepsilon}\ \lambda_{h:\gamma\to\Delta}$$
$$(g\,(\lambda_p\,\lambda_{q'}\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),(q'\,y))))\!:\!\Gamma\ \in\ R \qquad \text{(Ha)}$$

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

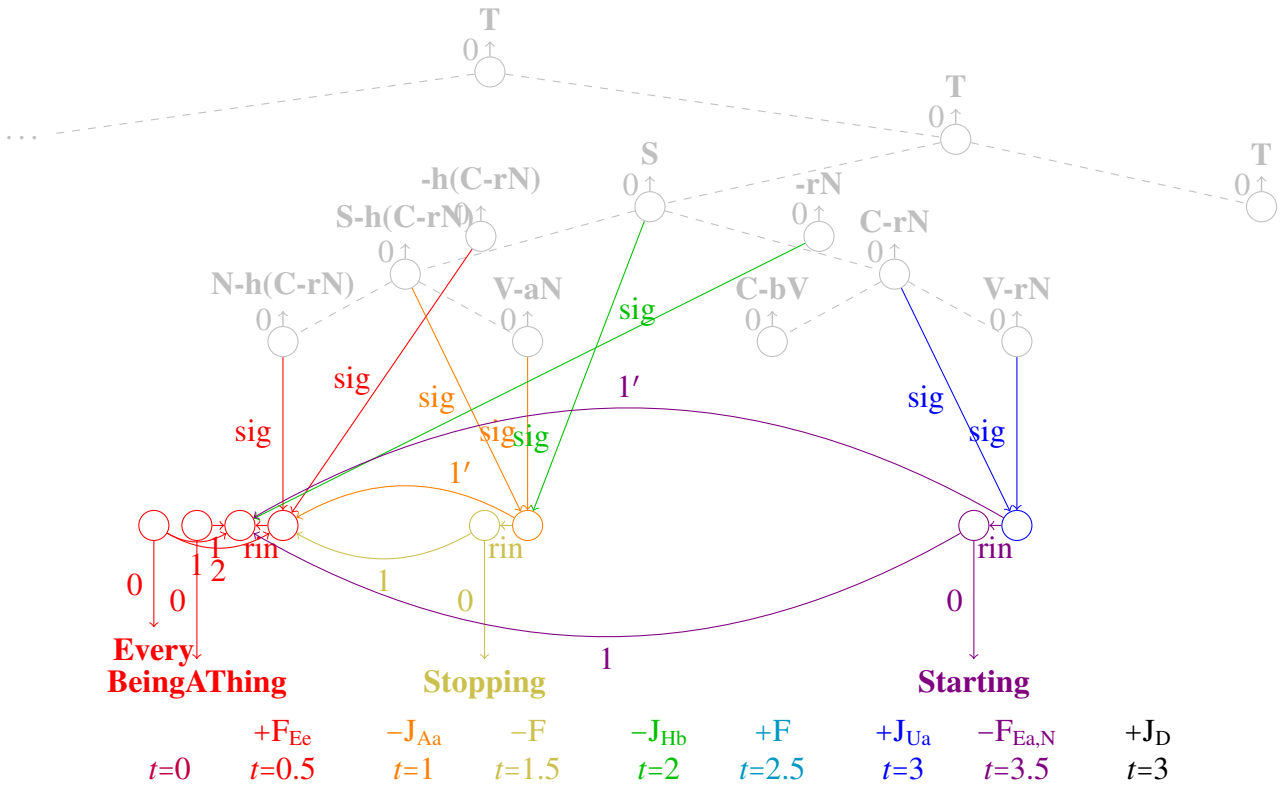
Ha:



Hb:



Example: '*Everything stops that starts.*'



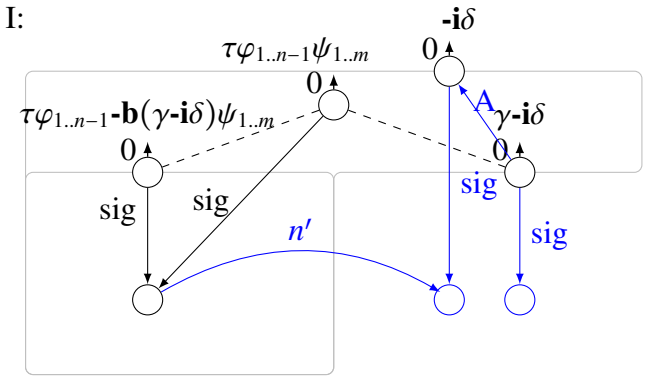
4.7.10 Interrogative clause attachment (binary)

We need to support arguments that become non-local dependencies.

This rule supports analyses for embedded questions, tough constructions, and free relatives:

(15) a. We [$_{\textbf{V-aN-b(V-iN)}}$ wonder] [$_{\textbf{V-iN}}$ [$_{\textbf{N-iN}}$ which pump] [$_{\textbf{V-gN}}$ Kim checked]].
b. Those pumps are [$_{\textbf{A-aN}}$ [$_{\textbf{A-aN-b(I-aN-gN)}}$ easy] [$_{\textbf{I-aN-gN}}$ to check]].
c. [$_{\textbf{N}}$ [$_{\textbf{N-b(V-gN)}}$ What] [$_{\textbf{V-gN}}$ Kim checked]] was the pump.

$$\lambda_{g:(\tau\varphi_{1..n-1}\textbf{-b}(\gamma\textbf{-i}\delta)\psi_{1..m}\to\Delta)\to\Gamma}\ \lambda_{q:\gamma\textbf{-i}\delta}\ \lambda_{q':\textbf{-i}\delta^s}\ \lambda_{h:\tau\varphi_{1..n-1}\psi_{1..m}\to\Delta}$$
$$(g\,(\lambda_p\,h\,(\lambda_x\,\exists_y\,(p\,x),(q\,y),(q'\,(\mathbf{f}_{n'}\,x))))):\Gamma\ \in\ R \qquad \text{(I)}$$



4.7.11 Type-change (unary)

We need type-changing rules to support bare relatives, *that* relatives, and full relatives:

(16) a. Something [C-rN [V-gN everyone likes]] is here.
b. Something [C-rN [C-gN that everyone likes]] is here.
c. Something [C-rN [V-rN which everyone likes]] is here.

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

(17) Every plan [C-rN [C-rN that the board approves] and [C-rN which is on the list]] is usable.

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

(18) * Every plan [? [C-rN that the board approves] and [A-aN on the list]] is usable.

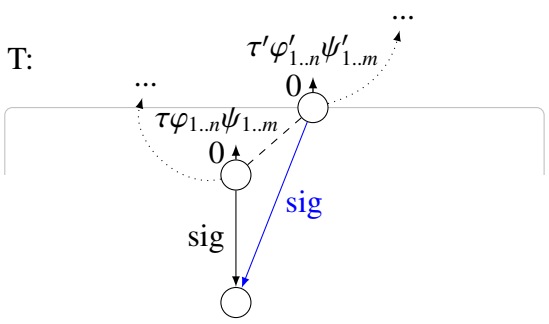
These rules also allow sentence types to be converted:

(19) 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]]!

Rules:

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

Graphically:



4.7.12 Relative clause attachment (binary)

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

(20) That's the person [N-rN a story [A-aN-rN about [N-rN [D-rN whose] life]]] was in the paper.

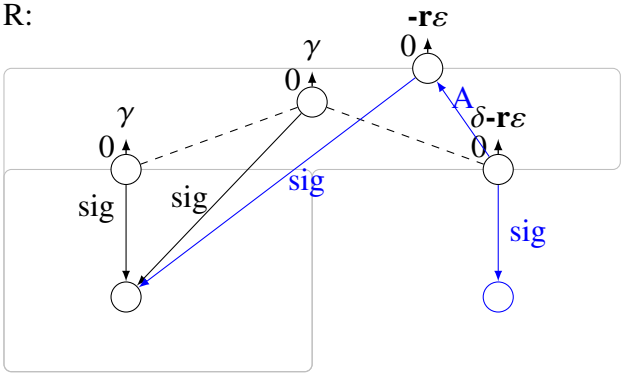
including those introducing quantifiers over the referent of a relative pronoun:

(21) There were ten pumps [N-rN half of which] were damaged.

Rules:

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

Graphically:



Restrictive and non-restrictive relative clauses can be distinguished lexically:

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

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

or using a type-changing rule:

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

4.7.13 Subject-auxiliary inversion (unary)

We need rules to support subject-auxiliary inverted questions.

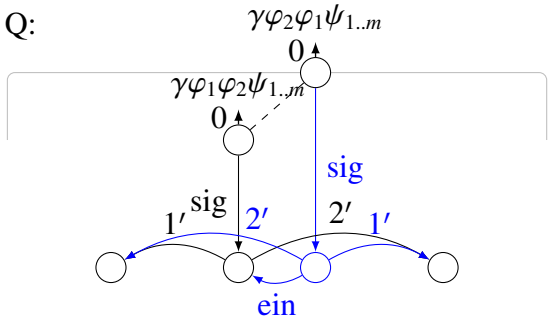
Subject-auxiliary inversion can also occur in declarative contexts:

(22) There/here/somewhere [$_{\textbf{V-g(A-aN)}}$ [$_{\textbf{Q-g(A-aN)}}$ are two pumps]].

Rules:

$$\lambda_{g:(\gamma\varphi_1\varphi_2\psi_{1..m}\to\Delta)\to\Gamma}\ \lambda_{h:\gamma\varphi_2\varphi_1\psi_{1..m}\to\Delta}$$
$$(g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),(\mathbf{f_{2'}}\,x){=}(\mathbf{f_{1'}}\,y),(\mathbf{f_{1'}}\,x){=}(\mathbf{f_{2'}}\,y),(\mathbf{f_{ein}}\,y){=}x)))\!:\Gamma\ \in\ R \qquad \text{(Q)}$$

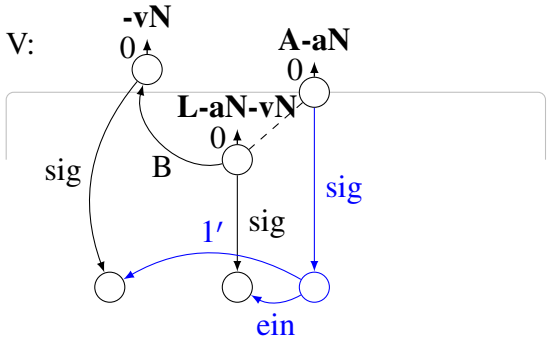
Graphically:



4.7.14 Passive attachment (unary)

We also need rules for passives:

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



4.7.15 Zero-head introduction (unary)

Finally, we need some rules to fill in empty heads.

This supports analyses of predicative noun phrases and time noun phrases:

(23) a. We considered the plan [$_{\mathbf{A\text{-}aN}}$ [$_{\mathbf{N}}$ a good suggestion]].
b. We traveled [$_{\mathbf{R\text{-}aN}}$ [$_{\mathbf{N}}$ the week before we moved]].

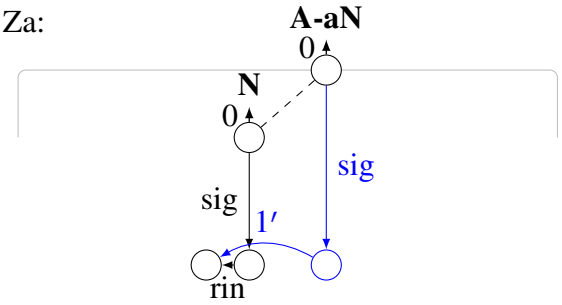
Rules:

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

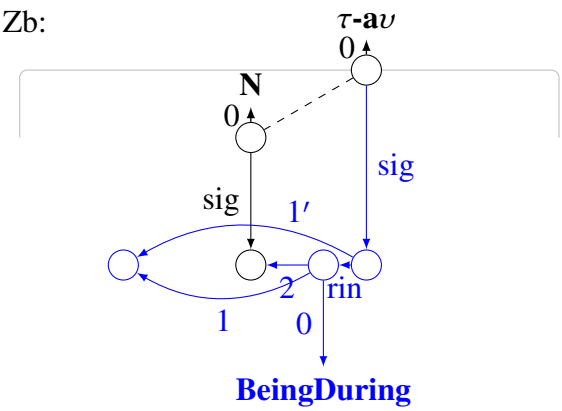
$$\lambda_{g:(\mathbf{N}\to\Delta)\to\Gamma}\ \lambda_{h:\tau\text{-}\mathbf{a}\upsilon\to\Delta}\ (g\,(\lambda_p\,h\,(\lambda_y\,\exists_x\,(p\,x),\ (\mathbf{f}_0\circ\mathbf{f}_{\mathbf{rin}}\,y)=\mathbf{BeingDuring},(\mathbf{f}_1\circ\mathbf{f}_{\mathbf{rin}}\,y)=(\mathbf{f}_{1'}\,y),(\mathbf{f}_2\circ\mathbf{f}_{\mathbf{rin}}\,y)=x))))\!:\!\Gamma\ \in\ R \qquad \text{(Zb)}$$

Graphically:

Za:



Zb:



References

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.

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

Botvinick, M. (2007). Multilevel structure in behavior and in the brain: a computational model of Fuster's hierarchy. *Philosophical Transactions of the Royal Society, Series B: Biological Sciences*, 362:1615–1626.

Church, A. (1940). A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2):56–68.

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

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.

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

Schuler, W., AbdelRahman, S., Miller, T., and Schwartz, L. (2010). Broad-coverage incremental parsing using human-like memory constraints. *Computational Linguistics*, 36(1):1–30.