

Lexical Precompilation by Cut-Elimination

Glyn Morrill

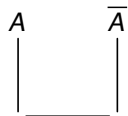
Universitat Politècnica de Catalunya

16th March 2019

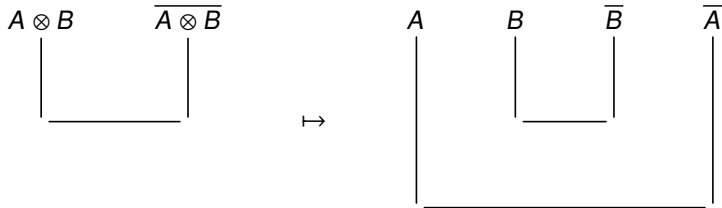
Communication includes the transmission of reasoning such as the validity of a syllogistic argument like:

Every man is mortal.
Socrates is a man.
Socrates is mortal.

Proof nets are a graphical proof representation, deeper than sequent calculus; e.g. for *Cut*:

$$\frac{\Gamma \Rightarrow A \quad \Delta(A) \Rightarrow B}{\Delta(\Gamma) \Rightarrow B} \text{Cut}$$


Computation can be seen as Cut-elimination. Cut-elimination in proof nets is geometrical interaction such as:



The computational metaphor in cognitive science already heralds communication as computation.

In this presentation we profess more specifically lexical precompilation by Cut-elimination.

Outline

In logical categorial grammar words and expressions are classified by types according to their syntax and semantics, and grammaticality/meaningfulness is determined by whether there is a proof in a type calculus which is universal.

Connectives:

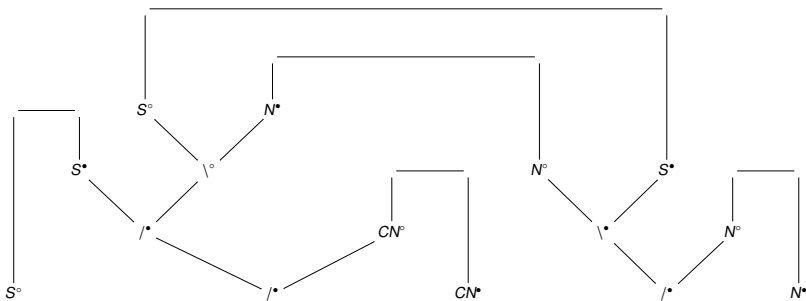
	cont. mult.	disc. mult.	add.	qu.	norm. mod.	brack. mod.	subexp.	limited contr. & weak.
primary	/ . 	$\uparrow$ $\odot$ J $\downarrow$	& +	$\wedge$ $\vee$	$\square$ $\diamond$	$\langle \rangle$ $[]^{-1}$	! ?	 W
sem. inactive variants	$\bullet \dashv \circ$ $\circ \dashv \bullet$ $\bullet$ $\circ$	$\uparrow \circ$ $\circ \uparrow$ $\bullet$ $\circ$	$\sqcap$ $\sqcup$	$\forall$ $\exists$	$\blacksquare$ $\blacklozenge$			
det. synth.	$\triangleleft^{-1}$ $\triangleleft$	$\triangleright^{-1}$ $\triangleright$	$\sim$ $\wedge$					
nondet. synth.	$\div$ $\times$	$\Uparrow$ $\Downarrow$ $\odot$						

- ▶ Logical categorial grammar reduces grammaticality to provability in a categorial logic.
- ▶ Consequently, in logical categorial grammar:
 - a) parsing is deduction; and
 - b) syntactic structures/parse structures are proofs.

- ▶ But what is a proof?
- ▶ E.g. sequent proofs are not canonical (spurious ambiguity/derivational equivalence):

$$\begin{array}{c}
 \vdots \\
 \frac{N, (N \setminus S)/N, N \Rightarrow S}{(N \setminus S)/N, N \Rightarrow N \setminus S} \backslash R \\
 \frac{CN \Rightarrow CN \quad S/(N \setminus S), (N \setminus S)/N, N \Rightarrow S}{(S/(N \setminus S))/CN, CN, (N \setminus S)/N, N \Rightarrow S} /L
 \end{array}
 \quad
 \begin{array}{c}
 \vdots \\
 \frac{CN \Rightarrow CN \quad S/(N \setminus S), N \setminus S \Rightarrow S}{(S/(N \setminus S))/CN, CN, N \setminus S \Rightarrow S} /L \\
 \frac{N \Rightarrow N \quad (S/(N \setminus S))/CN, CN, N \setminus S \Rightarrow S}{(S/(N \setminus S))/CN, CN, (N \setminus S)/N, N \Rightarrow S} /L
 \end{array}$$

- ▶ Girard's (Cut-free) proof nets are a good answer.



- ▶ These are graphs which must satisfy certain local and global criteria to be correct as proofs (proof nets).

In this talk

- ▶ We consider continuous and discontinuous types in relation to centre-embedding and quantifier scoping, and proof nets as a parallelized model of their incremental processing;
- ▶ We represent semantics as well as syntax by proof nets and words by partial proof nets or: *modules* (Ph. de Groote, A. Lecomte & C. Retoré);
- ▶ From lexico-syntactic evaluation as Cut-elimination between lexical and compositional semantics, we observe that this lexico-syntactic geometrical interaction can be partially precomputed by Cut-elimination within words, compiling them systematically into modules.

Continuous Types

Continuous syntactical algebra:

- a. Let V be a set of vocabulary items.
- b. Let $L = V^*$ be the set of all strings over V .
- c. Let 0 be the empty string.
- d. Let $+$ be the operation of concatenation of strings.
- e. Then L is *closed* under $+$: the concatenation of any two strings over V is another string over V .
- f. And $(s_1 + s_2) + s_3 = s_1 + (s_2 + s_3)$ (associativity) and $0 + s = s = s + 0$ (neutral element).
- g. We call $(L, +, 0)$ a *continuous syntactical algebra*, which technically is a monoid.

Basic types $\mathcal{A}$

<i>S</i>	(declarative) sentence	<i>John loves Mary</i>
<i>N</i>	(referring) nominal	<i>the man, John</i>
<i>CN</i>	count noun	<i>man, man that walks, man that Mary loves</i>
<i>PP</i>	prepositional phrase	<i>to Mary</i>
<i>CP</i>	complementizer phrase	<i>that John laughs</i>
$\vdots$		

Type formulas $\mathcal{F}$

$\mathcal{F} ::= \mathcal{A}$

$\mathcal{F} ::= I$ continuous unit $[[I]] = \{0\}$

$\mathcal{F} ::= \mathcal{F}/\mathcal{F}$ prefix $[[C/B]] = \{s_1 \mid \text{for all } s_2 \in [[B]], s_1 + s_2 \in [[C]]\}$

$\mathcal{F} ::= \mathcal{F} \backslash \mathcal{F}$ suffix $[[A \backslash C]] = \{s_2 \mid \text{for all } s_1 \in [[A]], s_1 + s_2 \in [[C]]\}$

$\mathcal{F} ::= \mathcal{F} \cdot \mathcal{F}$ conc $[[A \cdot B]] = \{s_1 + s_2 \mid s_1 \in [[A]] \ \& \ s_2 \in [[B]]\}$

Sequent:

A sequent $A_1, \dots, A_n \Rightarrow A$ comprises an *antecedent* list of types $A_1, \dots, A_n$ and a *succedent* type A ; it is *valid* if and only if $[[A_1 \cdots A_n]] \subseteq [[A]]$ in all syntactical interpretations.

Example lexical assignments

John: $N: j$

loves: $(N \backslash S) / N: \text{love}$

man: $CN: \text{man}$

Mary: $N: m$

that: $(CN \backslash CN) / (N \backslash S): \lambda x \lambda y \lambda z ((\wedge (y \ z)) (x \ z))$

that: $(CN \backslash CN) / (S / N): \lambda x \lambda y \lambda z ((\wedge (y \ z)) (x \ z))$

the: $N / CN: \iota$

walks: $N \backslash S: \text{walk}$

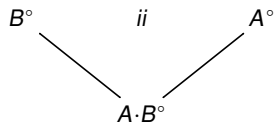
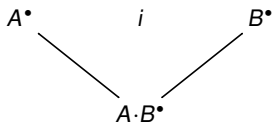
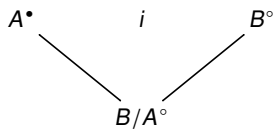
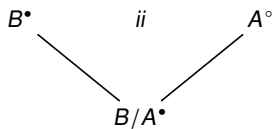
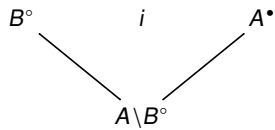
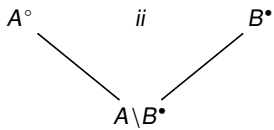
(Continuous) syntactic structures

Polar type:

A *polar type* A^p comprises a type A and a polarity $p = \bullet$ (input) or $\circ$ (output); where L is a polar type, $\bar{L}$ is the same type with the opposite polarity.

Polar type tree:

The *polar type tree* of a polar type A^p is the ordered tree that results from unfolding A^p up to atomic leaves as follows:



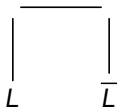
Proof frame:

The *proof frame* of a sequent $A_1, \dots, A_n \Rightarrow A$ is the sequence of polar type trees:

$$\langle A^\circ, A_1^\bullet, \dots, A_n^\bullet \rangle$$

Proof structure:

A *proof structure* is the result of connecting (by an *axiom link*) every leaf L in a proof frame to exactly one complementary leaf $\bar{L}$.



Proof net (syntactic structure)

A *proof net* is a proof structure which satisfies:

- a. *Planarity*: The axiom links are planar, i.e. they can be drawn in the halfplane without crossing lines.
- b. *1-acyclicity*: Every cycle crosses both edges of some *i*-link.

Theorem (*soundness and completeness*)

A sequent is valid if and only if there is a proof net over its frame.

Proof. Pentus (1998) for completeness of sequent calculus with respect to monoids. Girard (1987), Danos and Regnier (1989), Roorda (1991) and Retoré (1996) for proof nets and necessity and sufficiency of the correctness criteria.

Continuous incremental parsing model

Morrill (2000; 2011, ch. 4). We illustrate the idea with the processing of this sentence:

John loves Mary.

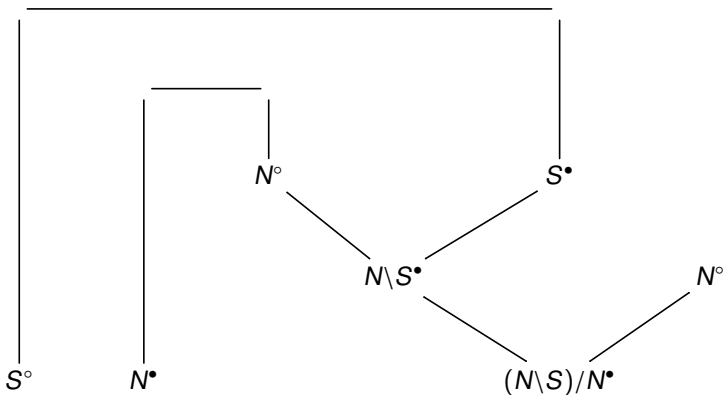
Initially, a sentence is being sought, and after hearing the first word its type is given:

S°

$N^\bullet$

John

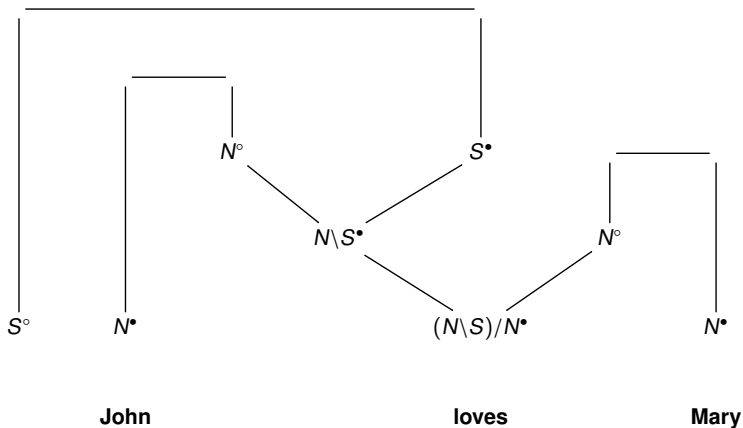
When the second word **loves** is heard, its unfolded type is connected by two dependencies: the subject sought is given by the first word **John** and the sentence projected is sought by the initial expectation of a sentence. We represent this as follows:



John

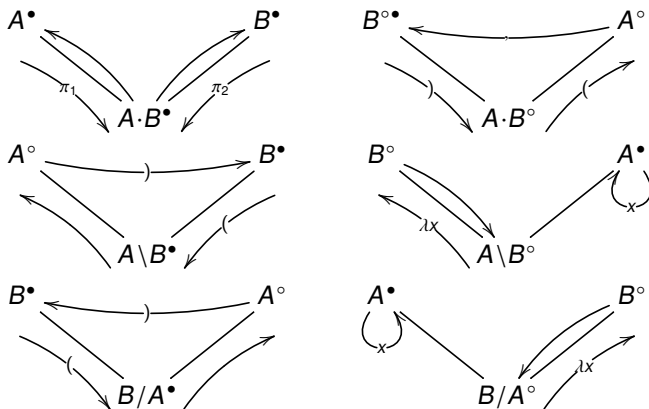
loves

When the final word **Mary** is heard, the parse is completed thus:



Semantic trip (de Groote and Retoré (1996))

With each $\backslash^\circ$ and $/^\circ$ logical link assigned a distinct semantic variable, the *semantic trip* starts upwards at the unique open output root (origin) and generates a textual form thus, bouncing with the associated semantic form at input roots:

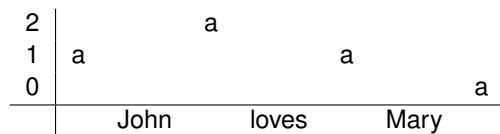


A semantic trip ends when it returns to the origin, having visited each node twice, once upwards and once downwards.

$((love\ m)\ j)$

Complexity metric

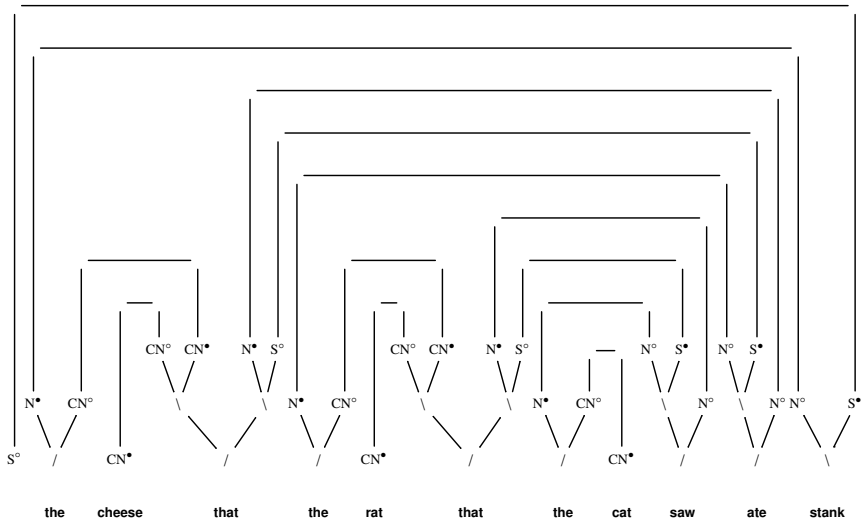
Our measure of complexity arises from the number of unresolved valencies at each word boundary in incremental processing: the load on working memory as measured by the number of items that must be kept on the stack at each point (Gibson 1998, Johnson 1998, Morrill 2000). Hence for our example the complexity profile is as follows:



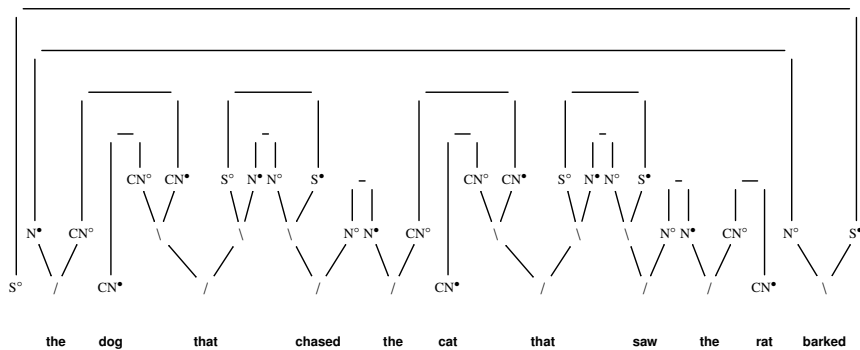
Centre-embedding

Nested subject and object relatives are analysed, under the model outlined, as follows.

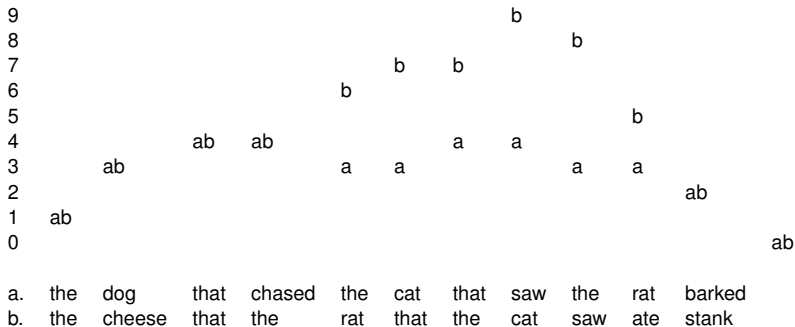
Centre-embedding relativisation



Non-centre-embedding relativisation



Let us compare the complexity profiles of the two sentences:



The complexity profiles reflect the relative difficulty of centre embedding as compared to non-centre embedding relativisation.

Discontinuous Types

- a. gave ... the cold shoulder. 'shun'
- b. put ... on the spot. 'embarrass'

Discontinuous syntactical algebra

- a. Let V be a set of vocabulary items.
- b. Let alphabet $\Sigma = V \cup \{1\}$, where the *placeholder* $1 \notin V$.
- c. The *sort* $\sigma(s)$ of a string $s \in \Sigma^*$ is the number of 1's it contains.
- d. The domains L_i of strings of sort i are defined by:

$$L_i = \{s \in \Sigma^* \mid \sigma(s) = i\}$$

- e. We define the operation $+$ of concatenation as in a continuous syntactical algebra; its functionality is $+: L_i, L_j \rightarrow L_{i+j}$.
- f. We define the operations $\times_k, k \in \{>, <\}$ of intercalation as the result of replacing by the second operand the leftmost ($>$) or rightmost ($<$) placeholder in the first operand; their functionality is $\times_k; L_{i+1}, L_j \rightarrow L_{i+j}$.
- g. A *discontinuous syntactical algebra* is

$$(\{L_i\}_{i \in \mathbb{N}}, +, \{\times_k\}_{k \in \{>, <\}}, 0, 1)$$

Basic types $\mathcal{A}_0$ of sort 0: $S, N, \dots$ Basic types $\mathcal{A}_1$ of sort 1: $Si, \dots$

Sorted type formulas $\mathcal{F}_i$:

$\mathcal{F}_i ::= \mathcal{A}_i$		
$\mathcal{F}_0 ::= I$	continuous unit	$[[I]] = \{0\}$
$\mathcal{F}_i ::= \mathcal{F}_{i+j}/\mathcal{F}_j$	prefix	$[[C/B]] = \{s_1 \mid \text{for all } s_2 \in [[B]], s_1 + s_2 \in [[C]]\}$
$\mathcal{F}_j ::= \mathcal{F}_i \backslash \mathcal{F}_{i+j}$	sufix	$[[A \backslash C]] = \{s_2 \mid \text{for all } s_1 \in [[A]], s_1 + s_2 \in [[C]]\}$
$\mathcal{F}_{i+j} ::= \mathcal{F}_i \cdot \mathcal{F}_j$	conc	$[[A \cdot B]] = \{s_1 + s_2 \mid s_1 \in [[A]] \ \& \ s_2 \in [[B]]\}$
$\mathcal{F}_1 ::= J$	discontinuous unit	$[[J]] = \{1\}$
$\mathcal{F}_{i+1} ::= \mathcal{F}_{i+j} \uparrow_k \mathcal{F}_j$	circumfix	$[[C \uparrow_k B]] = \{s_1 \mid \text{for all } s_2 \in [[B]], s_1 \times_k s_2 \in [[C]]\}$
$\mathcal{F}_j ::= \mathcal{F}_{i+1} \downarrow_k \mathcal{F}_{i+j}$	infix	$[[A \downarrow_k C]] = \{s_2 \mid \text{for all } s_1 \in [[A]], s_1 \times_k s_2 \in [[C]]\}$
$\mathcal{F}_{i+j} ::= \mathcal{F}_{i+1} \odot_k \mathcal{F}_j$	wrap	$[[A \odot_k B]] = \{s_1 \times_k s_2 \mid s_1 \in [[A]] \ \& \ s_2 \in [[B]]\}$

(Where the placeholder is unique the subscript $>$ or $<$ is omitted.)

Discontinuous polar type trees:

$$\begin{array}{ccc}
 A^\circ & & B^\bullet \\
 & \searrow \quad \nearrow & \\
 & A \downarrow_k B^\bullet &
 \end{array}
 \quad ii$$

$$\begin{array}{ccc}
 B^\circ & & A^\bullet \\
 & \searrow \quad \nearrow & \\
 & A \downarrow_k B^\circ &
 \end{array}
 \quad i$$

$$\begin{array}{ccc}
 B^\bullet & & A^\circ \\
 & \searrow \quad \nearrow & \\
 & B \uparrow_k A^\bullet &
 \end{array}
 \quad ii$$

$$\begin{array}{ccc}
 A^\bullet & & B^\circ \\
 & \searrow \quad \nearrow & \\
 & B \uparrow_k A^\circ &
 \end{array}
 \quad i$$

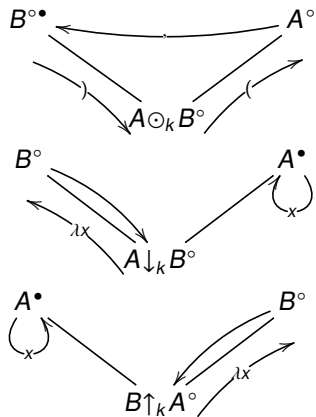
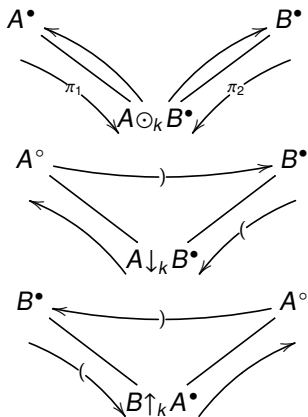
$$\begin{array}{ccc}
 A^\bullet & & B^\bullet \\
 & \searrow \quad \nearrow & \\
 & A \odot_k B^\bullet &
 \end{array}
 \quad i$$

$$\begin{array}{ccc}
 B^\circ & & A^\circ \\
 & \searrow \quad \nearrow & \\
 & A \odot_k B^\circ &
 \end{array}
 \quad ii$$

Correctness criteria:

Planarity needs to be relaxed. 1-acyclicity is necessary but not sufficient for correctness. Morrill and Fadda (2008), Fadda (2010), Moot (2016), but the precise nature of the partial planarity required is an open question.

Discontinuous semantic trip



Quantifier scoping

Let us consider quantifier scoping and quantifier scope preference. Let there be lexical assignments as follows:

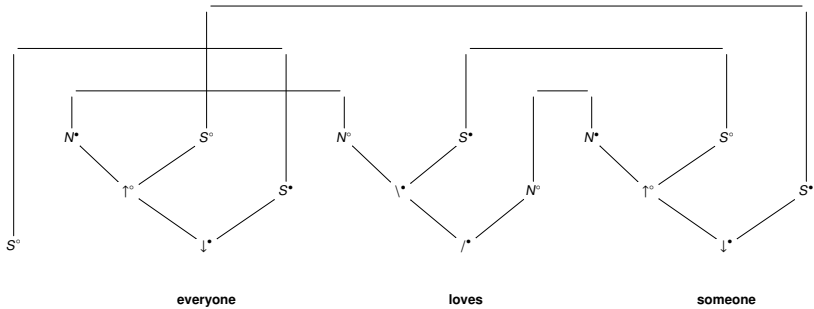
everyone: $(S \uparrow N) \downarrow S: \lambda x (\forall \lambda y ((\rightarrow (\text{person } y)) (x \ y)))$

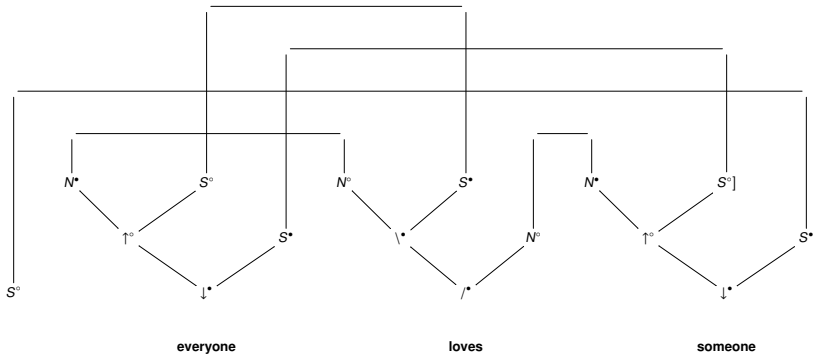
loves: $(N \setminus S) / N: \text{love}$

someone: $(S \uparrow N) \downarrow S: \lambda x (\exists \lambda y ((\wedge (\text{person } y)) (x \ y)))$

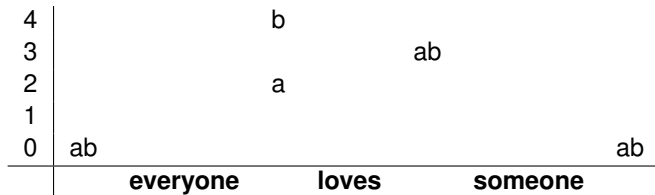
Such assignments allow quantifier phrases to occupy nominal positions while taking semantic scope sententially. For *Everyone loves someone* there are the following proof net analyses with the subject wide scope semantics (a) and the object wide scope semantics (b) respectively.

- a. $(\forall \lambda x((\rightarrow (\text{person } x)) (\exists \lambda y((\wedge (\text{person } y)) ((\text{love } y) x))))))$
- b. $(\exists \lambda y((\wedge (\text{person } y)) (\forall \lambda x((\rightarrow (\text{person } x)) ((\text{love } y) x))))))$





For the subject wide scope analysis with semantics a and the object wide scope analysis with semantics b the complexity profiles are as follows:



The variation in the complexity profiles accords with the left-to-right quantifier scope preference.

Lexico-Syntactic Cut-elimination

De Groote and Retoré (1996) show how semantics as well as syntax can be represented in the formalism of proof nets. They consider the following points:

- ▶ that lexical semantics may be represented by an intuitionistic proof net;
- ▶ that the substitution of lexical semantics into derivational semantics corresponds to connecting with a Cut link the output conclusion of the lexical semantics proof net to the root of the polar type tree of the lexical type, and that semantic evaluation corresponds to elimination of Cuts from the resulting proof net.

Lecomte and Retoré (1995) introduce the slogan ‘words as modules’ for the idea that what is associated with a word need not be just a type, but may be a partial proof net, or module.

Following on from these ideas, Morrill (1999) proposes to preevaluate lexical and syntactic interaction by eliminating Cuts, so far as possible, from lexical modules built by step (ii) above. This precompilation of the categorial lexicon allows partial execution of much, though not all, lexico-syntactic interaction.

We illustrate with reference to the following lexical assignments:

bag+end: $N: b$

frodo: $N: f$

in: $(S \setminus S)/N: in$

inhabits: $(N \setminus S)/N: \lambda x \lambda y ((in\ x) (live\ y))$

lives: $N \setminus S: live$

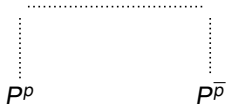
These define the following paraphrase:

a. **frodo+lives+in+bag+end**: $S: ((in\ b) (live\ f))$

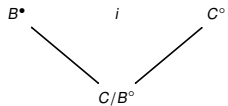
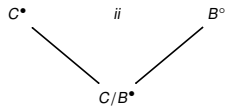
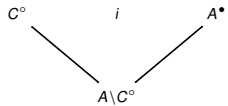
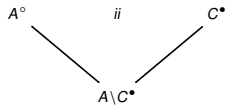
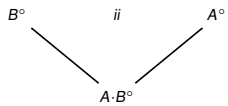
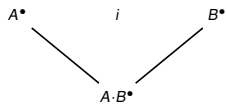
b. **frodo+inhabits+bag+end**: $S: ((in\ b) (live\ f))$

Syntactic Nets

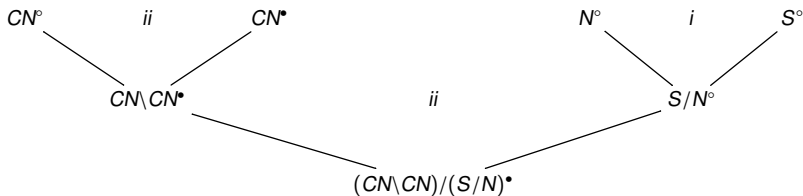
An *identity link* is of the form:



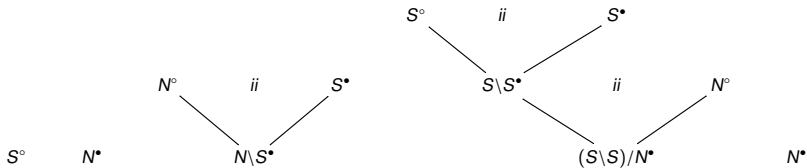
A *logical link* is one of the local trees:



A *polar type tree* is a tree the leaves of which are literals and each local tree of which is a logical link. For example, the prosodic tree for $(CN \setminus CN)/(S/N)^{\bullet}$ is:



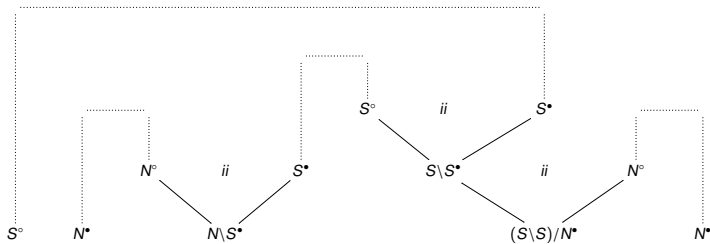
A *syntactic frame* is a cyclic list of polar type trees exactly one of which has an output root. For example:



A *syntactic net* is the result of connecting by an identity link every leaf in a syntactic frame with a unique complementary leaf such that:

- ▶ (*1-Acyclicity*). Every cycle crosses both edges of some i-link;
- ▶ (*Planarity*). The identity links are planar in the cyclic ordering.

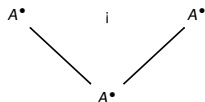
For example, the following is a syntactic net:



Semantic nets

We define partial proof structures for the Lambek calculus with Permutation and Contraction with Cut (**LPC**+Cut).

A *contraction link* is of the form:



We define i - and j -links as 1-links. A *Cut link* is of the form:

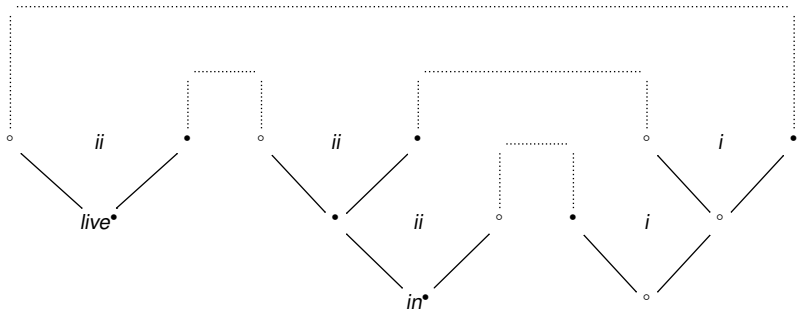


A *semantic polar type tree* is a tree the leaves of which are literals and each local tree of which is a logical link or a contraction link. Note that syntactic polar type trees are a special case of semantic polar type trees without contraction links. A *semantic frame* is a bag of semantic polar type trees.

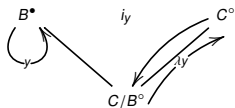
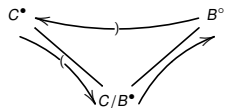
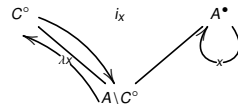
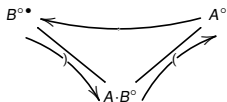
A *semantic module* is the result of i) connecting, in a semantic frame, some leaves with unique complementary leaves by an identity link, and some roots with unique complementary roots by a Cut link, and ii) associating semantic constants to every open input root, such that:

- ▶ (*Orientation*). There is a unique open output root.
- ▶ (*1-Acyclicity*). Every cycle crosses both edges of some 1-link.

A *semantic net* is a semantic module with no open leaves. For example, the following is a semantic net:



Semantic trip of a semantic net

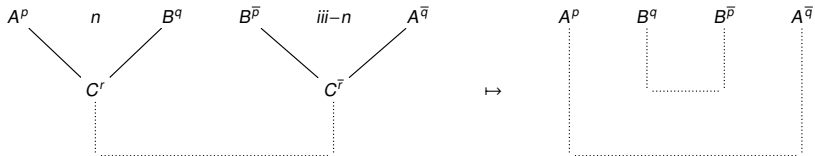


The *reading* of a semantic net is the semantic form generated by its semantic trip. For example, the semantic reading of our example is

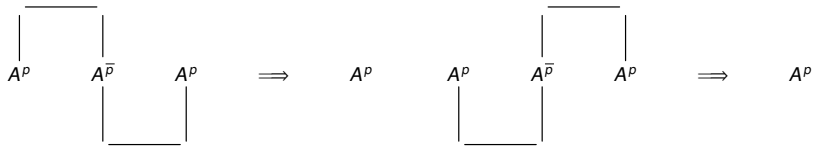
$$\lambda x \lambda y ((in\ x)\ (live\ y))$$

The following Cut-elimination conversions on semantic nets preserve readings:

Principal conversions:

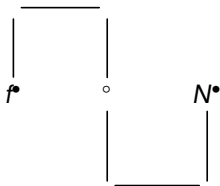


Snake conversions:



Let an *initial lexico-syntactic module* for a lexical assignment $\alpha: A: \phi$ be a module which results from connecting by a Cut link the syntactic polar type tree of $A^\bullet$ with a semantic net the reading of which is ϕ . The corresponding *final lexico-syntactic module* is the result of normalising, preserving the order on open leaves.

For example, for **Frodo**: $N: f$ we have the initial lexico-syntactic module:



Which simplifies by snake conversion to the final module:

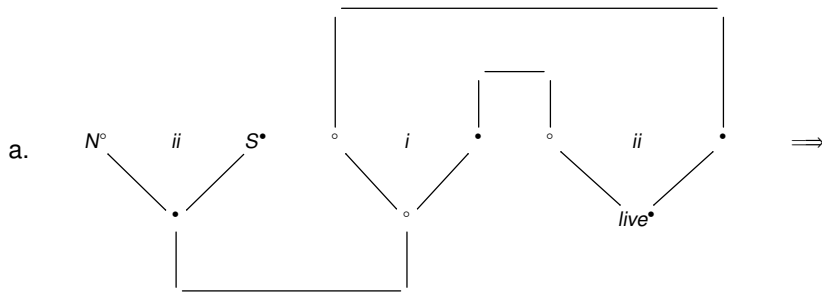
$N^\bullet$
 f

Similarly, for **bag+end**: $N: b$ we obtain the final module:

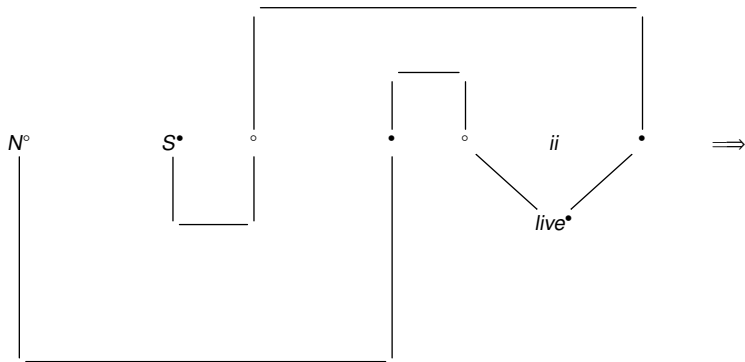
$N^\bullet$
 b

Cut-elimination preevaluation

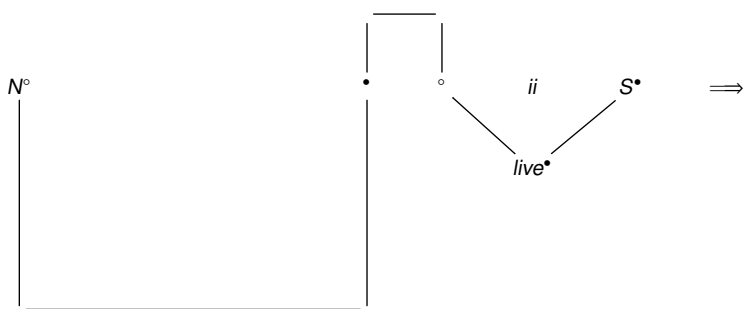
For **lives**: $N \setminus S$: *live* we have the initial module, Cut-elimination, and final module:



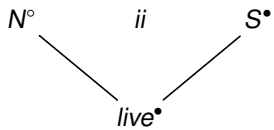
b.



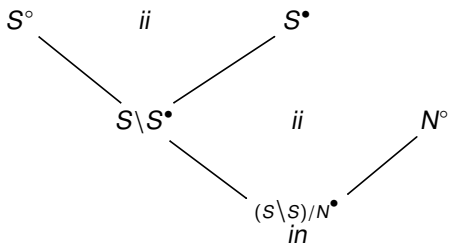
C.



d.

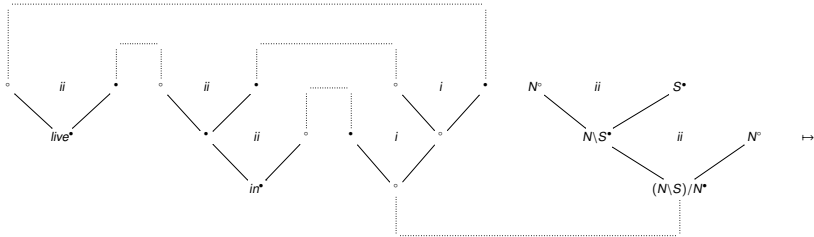


Similarly, for **in**: $(S \setminus S)/N$: *in* we obtain the final module:

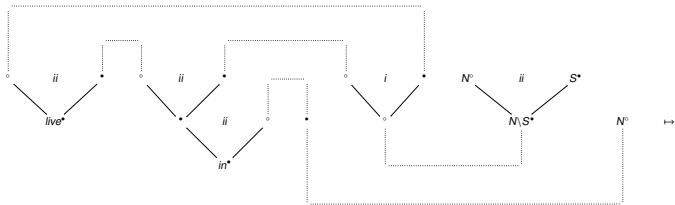


For **inhabits**: $(N \setminus S)/N$: $\lambda x \lambda y ((in\ x) (live\ y))$ we have the initial module, preevaluation, and final module:

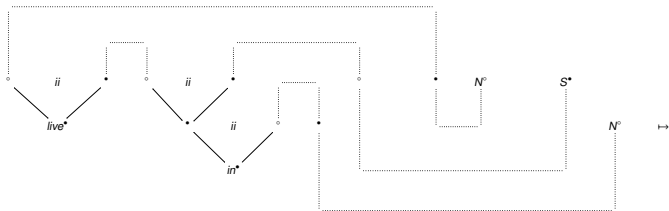
a.



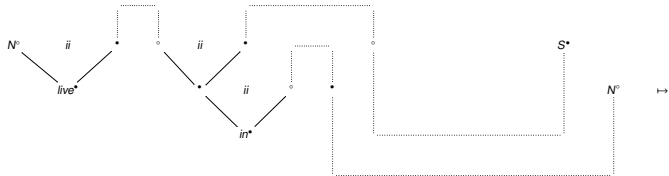
b.



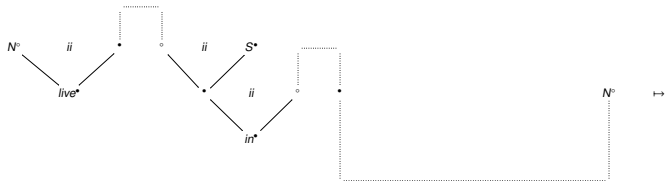
c.



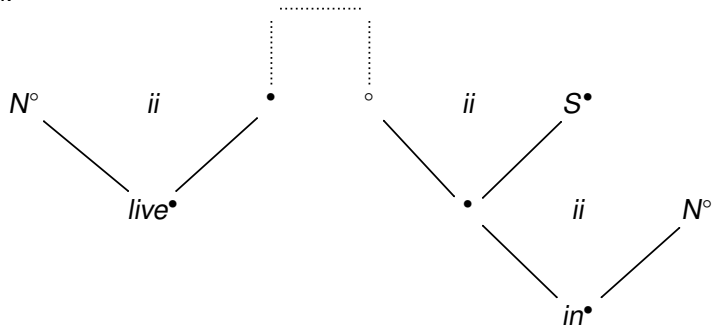
d.



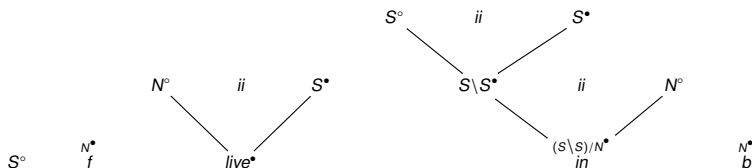
e.



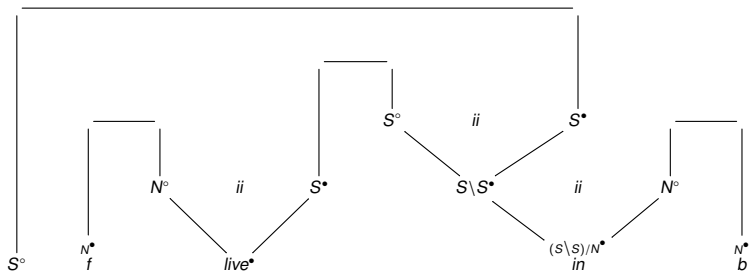
f.



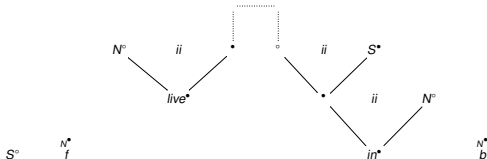
So, for example, from the final lexico-syntactic modules we have for the sentence *Frodo lives in Bag End* the lexico-syntactic frame:



And the lexico-syntactic net:



And for *Frodo inhabits Bag End* from the final lexico-syntactic modules we have the lexico-syntactic frame



and the same lexico-syntactic net again, capturing the paraphrase.

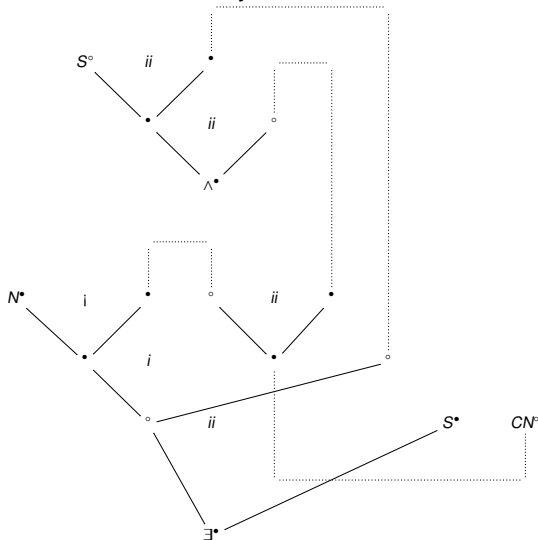
The shared semantic reading is $((in\ b)\ (live\ f))$.

Words

In this section we illustrate final lexico-syntactic modules for sample lexical assignments.

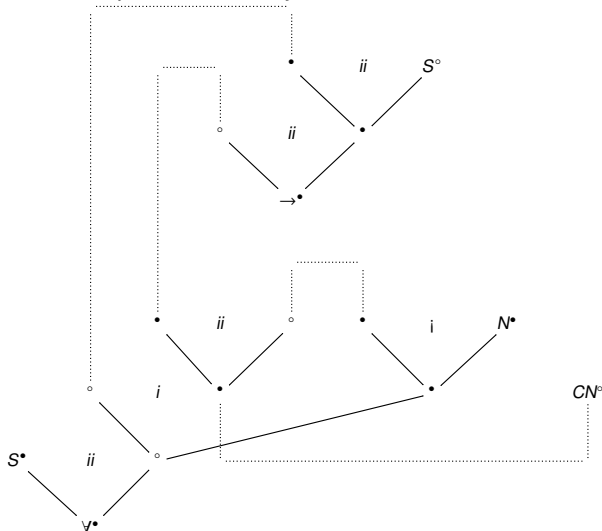
a: $\lambda x \lambda y (\exists \lambda z (\wedge (x z) (y z))) : ((S/N) \setminus S) / CN$

Indefinite article in subject.



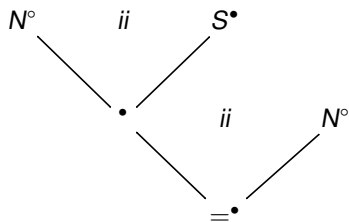
every: $(S/(N \setminus S))/CN: \lambda x \lambda y (\forall \lambda z (\rightarrow (x z) (y z)))$

Universal quantifier in subject.



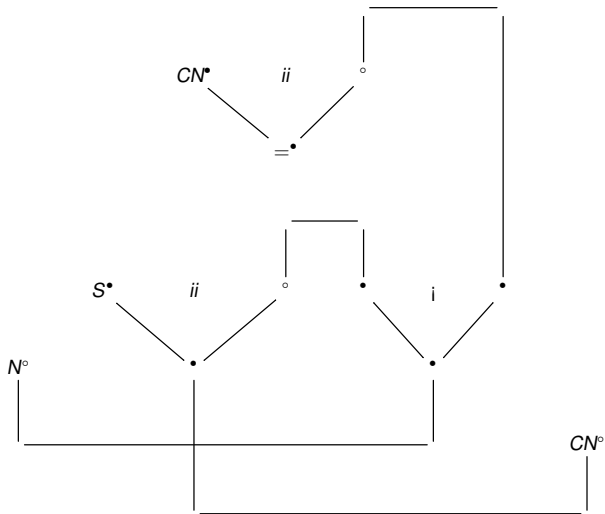
is: $(N \setminus S) / N: =$

Copula of identification



is: $(N \setminus S) / (CN / CN): \lambda x \lambda y (x (= y) y)$

Copula of predication



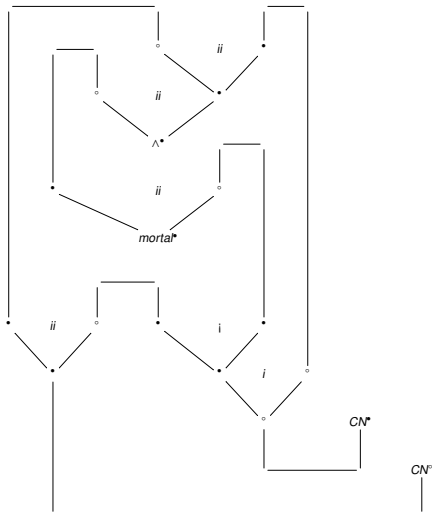
man: *CN: man*

Count noun

CN•
man

mortal: $CN/CN: \lambda x \lambda y (\wedge (mortal\ y) (x\ y))$

Intersective adjective



socrates: N : s

Proper name

$N^{\bullet}$
 S

Sentences

We analyse the classical syllogism:

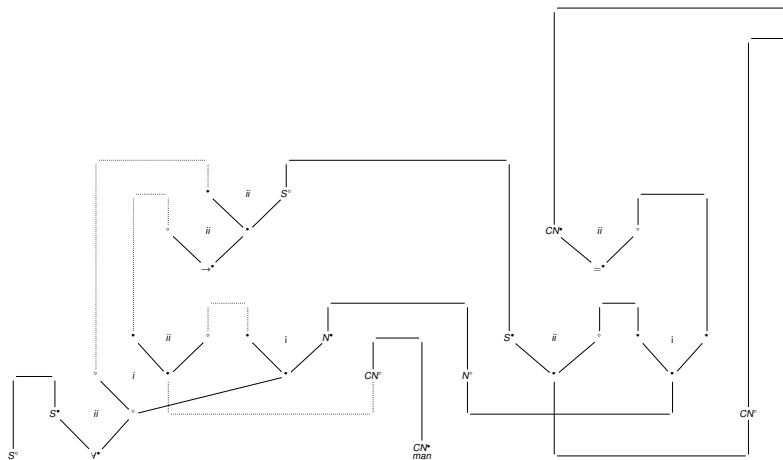
Every man is mortal.

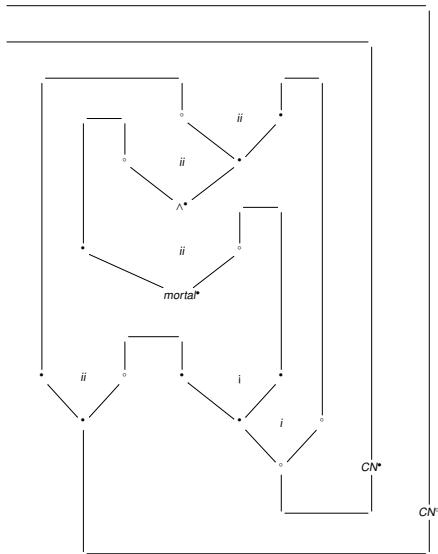
Socrates is a man.

Socrates is mortal.

Every man is mortal

The lexico-syntactic net for *Every man is mortal* is:



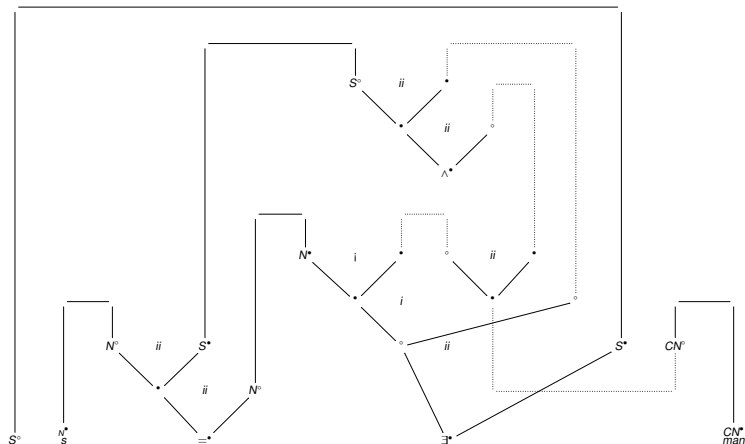


The semantic reading is:

$$\begin{aligned}
 & (\forall \lambda x ((\rightarrow (man\ x)) (\lambda y ((\wedge (mortal\ y)) ((= x\ y)) x)))) \equiv \\
 & (\forall \lambda x (\rightarrow (man\ x) (mortal\ x)))
 \end{aligned}$$

Socrates is a man

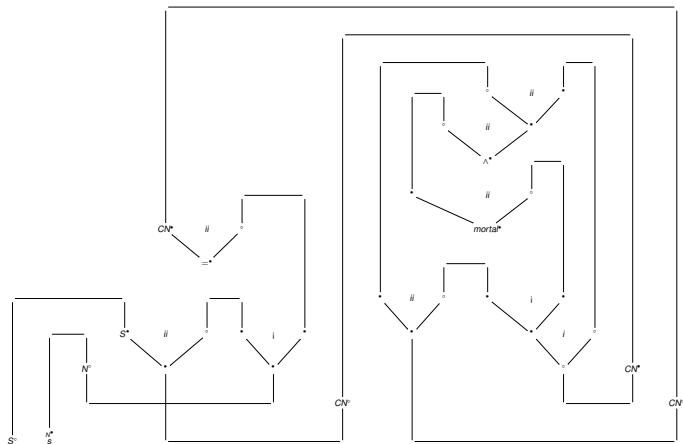
The lexico-syntactic net for *Every man is mortal* is:



The semantic reading is: $(\exists \lambda x((\wedge (man\ x)) ((= x) s)))) \equiv (man\ s)$

Socrates is mortal

The lexico-syntactic net is:



The semantic reading is: $(\lambda x((\wedge (mortal\ x)) ((= s)\ x))\ s) \equiv (mortal\ s)$

Indeed, according to our analyses the premises of the syllogism entail its conclusion.

Every man is mortal.

Socrates is a man.

Socrates is mortal.

$(\forall \lambda x(\rightarrow (man\ x) (mortal\ x)))$

$(man\ s)$

$(mortal\ s)$

Bibliography



J. M. Andreoli.

Logic programming with focusing in linear logic.

Journal of Logic and Computation, 2(3):297–347, 1992.



V. Danos and L. Regnier.

The structure of multiplicatives.

Archive for Mathematical Logic, 28:181–203, 1989.



P. de Groote and Ch. Retoré.

On the semantic readings of proof nets.

In G.-J. M. Kruijff, G. Morrill, and R. T. Oehrle, editors, *Proceedings of Formal Grammar*, pages 57–70, Prague, 1996.



Mario Fadda.

Geometry of Grammar: Exercises in Lambek Style.

PhD thesis, Universitat Politècnica de Catalunya, Barcelona, 2010.



E. Gibson.

Linguistic complexity: locality of syntactic dependencies.

Cognition, 68:1–76, 1998.



Jean-Yves Girard.

Linear logic.

Theoretical Computer Science, 50:1–102, 1987.



M.E. Johnson.

Proof nets and the complexity of processing center-embedded constructions.

Journal of Logic, Language, and Information, 4(7):433–447, 1998.



Joachim Lambek.

The mathematics of sentence structure.

American Mathematical Monthly, 65:154–170, 1958.

Reprinted in Buszkowski, Wojciech, Wojciech Marciszewski, and Johan van Benthem, editors, 1988, *Categorical Grammar*, Linguistic & Literary Studies in Eastern Europe volume 25, John Benjamins, Amsterdam, 153–172.



A. Lecomte and Ch. Retoré.

Words as modules and modules as partial proof-nets.

In V.M. Abrusci and C. Casadio, editors, *Proofs and Linguistic categories – Applications of Logic to the Analysis and Implementation of Natural Language*, Bologna, 1996. CLUEB.



Richard Moot.

Proof nets for the displacement calculus.

In Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald, and Sylvain Pogodalla, editors, *Formal Grammar: 20th and 21st International Conferences*, volume 9804 of *LNCS, FoLLI Publications in Logic, Language and Information*, pages 273–289, Berlin, 2016. Springer.



Glyn Morrill.

Geometry of Lexico-Syntactic Interaction.

In *Proceedings of the Ninth European Chapter of the Association for Computational Linguistics*, pages 61–70, Bergen, 1999.



Glyn Morrill.

Incremental Processing and Acceptability.

Computational Linguistics, 26(3):319–338, 2000.



Glyn Morrill and Mario Fadda.

Proof Nets for Basic Discontinuous Lambek Calculus.

Logic and Computation, 18(2):239–256, 2008.



Glyn V. Morrill.

Categorical Grammar: Logical Syntax, Semantics, and Processing.

Oxford University Press, New York and Oxford, 2011.



M. Pentus.

Free monoid completeness of the Lambek calculus allowing empty premises.

In J.M. Larrazabal, D. Lascar, and Mints G., editors, *Logic colloquium '96*, volume 12, pages 171–209. Springer, 1998.

Lecture notes in Logic.



Ch. Retoré.

Calcul de Lambek et Logique Linéaire.

Traitement automatique du langage, 2(37):39–70, 1995.



Dirk Roorda.

Resource Logics: Proof-Theoretical Investigations.

PhD thesis, Universiteit van Amsterdam, 1991.