Computational Linguistics: Syntax-Semantics Interface

Raffaella Bernardi

KRDB, Free University of Bozen-Bolzano

P.zza Domenicani, Room: 2.28, e-mail: bernardi@inf.unibz.it

Contents

1. Lambda terms and DCG

We will look at the compositional approach to the syntax-semantics interface and build the meaning representation in parallel to the syntactic tree. This reduces to have a **rule-to-rule** system, i.e. each syntactic rule correspond to a semantic rule.

Syntactic Rule 1 $S \rightarrow NP\ VP$

Semantic Rule 1 If the logical form of the NP is α and the logical form of the VP is β then the logical form for the S is $\beta(\alpha)$.

Syntactic Rule 2 $VP \rightarrow TV\ NP$

Semantic Rule 2 If the logical form of the TV is α and the logical form of the NP is β then the logical form for the VP is $\alpha(\beta)$.

1.1. Augumenting DCG with terms

That can also be abbreviated as below where γ, α and β are the meaning representations of S, NP and VP, respectively.

$$S(\gamma) \rightarrow NP(\alpha)\ VP(\beta)\ \gamma = \beta(\alpha)$$

This implies that lexical entries must now include semantic information. For instance, a way of writing this information is as below.

$$TV(\lambda x.\lambda y.wrote(y, x)) \rightarrow [wrote]$$

1.2. Quantified NP: terms and syntactic rules

We have seen that the term of a quantifier like "every student" is $\lambda Y.\forall z.Student(z) \rightarrow Y(z)$, which is of type $(e \rightarrow t) \rightarrow t$. Hence the sentence,

Every student left.

is obtained by applying the quantified noun phrase to the verb. In other words, if "Everybody" is of category NP we need the rule below:

$$S(\gamma) \rightarrow NP(\alpha)\ VP(\beta)\ \gamma = \alpha(\beta)$$

This has brought semanticists to change the meaning representation of the noun phrase too, since they have to be of the same "sort". E.g, "John" could be represented as

$$\lambda X.X(john)$$

a function of the same type as the quantified NP, i.e. $(e \rightarrow t) \rightarrow t$.

2. Categorial Grammar

- **Who:** Lesniewski (1929), Ajdukiewicz (1935), Bar-Hillel (1953).
- **Aim:** To build a language recognition device.
- **How:** Linguistic strings are seen as the result of concatenation obtained by means of syntactic rules starting from the categories assigned to lexical items. The grammar is known as Classical Categorial Grammar (CG).
- **Connection with Type Theory:** The syntax of type theory closely resembles the one of categorial grammar. The links between types (and lambda terms) with models, and types (and lambda terms) with syntactic categories, gives an interesting framework in which syntax and semantic are strictly related. (We will come back on this later.)

Categories: Given a set of basic categories ATOM, the set of categories CAT is the smallest set such that:

$$\mathsf{CAT} := \mathsf{ATOM} \mid \mathsf{CAT}\backslash\mathsf{CAT} \mid \mathsf{CAT}/\mathsf{CAT}$$

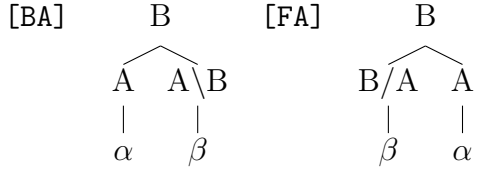
3. CG: Syntactic Rules

Categories can be composed by means of the syntactic rules below

`[BA]` If α is an expression of category A, and β is an expression of category $A\backslash B$, then $\alpha\beta$ is an expression of category B.

`[FA]` If α is an expression of category A, and β is an expression of category B/A, then $\beta\alpha$ is an expression of category B.

where `[FA]` and `[BA]` stand for Forward and Backward Application, respectively.



4. CG Lexicon: Toy Fragment

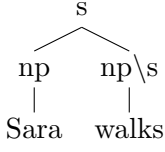
Let ATOM be $\{n, s, np\}$ (for nouns, sentences and noun phrases, respectively) and LEX as given below. Recall PSG rules: $np \to det\ n, s \to np\ vp, vp \to v\ np \ldots$

Lexicon

Sara	np	the	np/n
student	n	walks	$np\backslash s$
wrote	$(np\backslash s)/np$		

Sara walks $\in s$? $\rightsquigarrow$ $\underbrace{np}_{\text{Sara}}, \underbrace{np\backslash s}_{\text{walks}} \in s$? Yes

simply [BA]



5. Classical Categorial Grammar

Alternatively the rules can be thought of as Modus Ponens rules and can be written as below.

$$B/A, A \Rightarrow B \qquad \mathrm{MP_r}$$
$$A, A\backslash B \Rightarrow B \qquad \mathrm{MP_l}$$

$$\frac{B/A \quad A}{B}\ (\mathrm{MP_r}) \qquad \frac{A \quad A\backslash B}{B}\ (\mathrm{MP_l})$$

6. Classical Categorial Grammar. Examples

Given $\mathsf{ATOM} = \{np, s, n\}$, we can build the following lexicon:

Lexicon

John, Mary	$\in$	np	the	$\in$	np/n
student	$\in$	n			
walks	$\in$	$np\backslash s$			
sees	$\in$	$(np\backslash s)/np$			

Analysis

John walks $\in s$? $\rightsquigarrow$ $np, np\backslash s \Rightarrow s$? Yes

$$\frac{np \quad np\backslash s}{s}\,(\mathrm{MP_l})$$

John sees Mary $\in s$? $\rightsquigarrow$ $np, (np\backslash s)/np, np \Rightarrow s$? Yes

$$\frac{np \qquad \dfrac{(np\backslash s)/np \quad np}{np\backslash s}\,(\mathrm{MP_r})}{s}\,(\mathrm{MP_l})$$

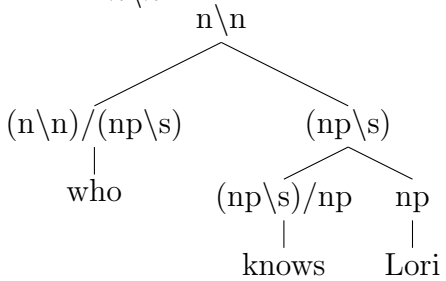
6.1. Relative Pronoun

Question Which would be the syntactic category of a relative pronoun in subject position? E.g. "the student **who** knows Lori"

[the [[student]$_n$ [who [knows Lori]$_{(np\backslash s)}$]$_?$]$_n$

who knows Lori $\in n\backslash n$? $\rightsquigarrow$ $(n\backslash n)/(np\backslash s), (np\backslash s)/np, np \Rightarrow n\backslash n$?

$$\dfrac{\dfrac{\text{who}}{(n\backslash n)/(np\backslash s)} \quad \dfrac{\dfrac{\text{knows}}{(np\backslash s)/np} \quad \dfrac{\text{Lori}}{np}\,(\text{MP}_\text{r})}{np\backslash s}}{n\backslash n}\,(\text{MP}_\text{r})$$



6.2. CFG and CG

Below is an example of a simple CFG and an equivalent CG:

CFG

```
S --> NP VP
VP --> TV NP
N --> Adj N

Lexicon:
Adj --> poor
NP --> john
TV --> kisses
```

CG Lexicon:

John:	np
kisses:	$(np\backslash s)/np$
poor:	n/n

7. CG: syntax-semantics interface

Summing up, CG specifies a language by describing the **combinatorial possibilities of its lexical items** directly, without the mediation of phrase-structure rules. Consequently, two grammars in the same system differ only in the lexicon.

The **close relation between the syntax and semantics** comes from the fact that the two syntactic rules are application of a functor category to its argument that corresponds to functional application of the lambda calculus.

We have to make sure that the lexical items are associated with **semantic terms** which correspond to the **syntactic categories**.

7.1. Mapping: types-categories

To set up the form-meaning correspondence, it is useful to build a language of semantic types in parallel to the syntactic type language.

Definition 7.1 (Types) Given a non-empty set of basic types Base, the set of types TYPE is the smallest set such that

i. $\mathsf{Base} \subseteq \mathsf{TYPE}$;
ii. $(a \rightarrow b) \in \mathsf{TYPE}$, if a and $b \in \mathsf{TYPE}$.

Note that this definition closely resembles the one of the syntactic categories of CG. The only difference is the lack of directionality of the functional type (a, b). A function mapping the syntactic categories into TYPE can be given as follows.

Definition 7.2 (Categories and Types) *Let us define a function* $\mathtt{type} : \mathit{CAT} \rightarrow \mathit{TYPE}$ *which maps syntactic categories to semantic types.*

$$\mathtt{type}(np) = e; \qquad \mathtt{type}(A/B) = (\mathtt{type}(B) \rightarrow \mathtt{type}(A));$$
$$\mathtt{type}(s) = t; \qquad \mathtt{type}(B\backslash A) = (\mathtt{type}(B) \rightarrow \mathtt{type}(A));$$
$$\mathtt{type}(n) = (e \rightarrow t).$$

7.2. CG: categories and terms

Modus ponens corresponds to functional application.

$$\frac{B/A : t \quad A : r}{B : t(r)}\ (\mathrm{MP_r}) \qquad \frac{A : r \quad A\backslash B : t}{B : t(r)}\ (\mathrm{MP_l})$$

Example

$$\frac{np : \mathtt{john} \quad np\backslash s : \mathtt{walk}}{s : \mathtt{walk(john)}}\ (\mathrm{MP_l})$$

$$np\backslash s : \lambda x.\mathtt{walk}(x) \quad (\lambda x.\mathtt{walk}(x))(\mathtt{john}) \rightsquigarrow_{\lambda-\mathrm{conv.}} \mathtt{walk(john)}$$

$$\frac{np : \mathtt{john} \qquad \dfrac{(np\backslash s)/np : \mathtt{know} \quad np : \mathtt{mary}}{np\backslash s : \mathtt{know(mary)}}\ (\mathrm{MP_r})}{s : \mathtt{know(mary)(john)}}\ (\mathrm{MP_l})$$

8. Logic Grammar

- **Aim:** To define the logic behind CG.
- **How:** Considering categories as formulae; $\backslash, /$ as logic connectives.
- **Who:** Jim Lambek [1958]

- **Proof Theory** Elimination and **Introduction** rules [Natural Deduction (ND) proof format]
- **Model Theory** (Kripke) Models. (if you don't what they does not matter and just think of Models for Prop. Logic)

Proof Theory ND is a proof system, i.e. a system to prove that some premises $\phi_1, \ldots \phi_n$ derive ($\vdash$) a conclusion (α). The proof consists of logical rules that do not consider the "meaning" (truth values) of the formulae involved rather their form (syntax). E.g. $A \rightarrow B, A \vdash B$

The system is proved to be sound and complete.

8.1. Natural Deduction

For each connective * there is a rule that says how we can **eliminate** it from the premises and how we can **introduce** it in the conclusion

$$\frac{premises}{conclusion}\ *$$

For instance, in Propositional Logic (PL), the elimination and introduction rules of $\wedge$ are:

$$\frac{A \wedge B}{A}\ \wedge \mathrm{E}_r \qquad \frac{A \quad B}{A \wedge B}\ \wedge \mathrm{I}$$

the elimination and introduction rules of $\rightarrow$ are:

$$\frac{A \rightarrow B \quad A}{B}\ \rightarrow \mathrm{E} \qquad \frac{\begin{array}{c}[A]^i \\ \vdots \\ B\end{array}}{A \rightarrow B}\ \rightarrow \mathrm{I}^i$$

8.2. Lambek Calculi

In the Lambek Calculus the connectives are $\backslash$ and $/$ (that behave like the $\rightarrow$ of PL except for their directionality aspect.)

Therefore, in the Lambek Calculus besides the elimination rules of $\backslash, /$ (that we saw in CG) we have their introduction rules.

$$\frac{B/A \quad A}{B}\ /\mathrm{E} \qquad \frac{A \quad A\backslash B}{B}\ \backslash\mathrm{E}$$

$$\frac{\begin{array}{c}[A]^i \\ \vdots \\ B\end{array}}{B/A}\ /\mathrm{I}^i \qquad \frac{\begin{array}{c}[A]^i \\ \vdots \\ B\end{array}}{A\backslash B}\ \backslash\mathrm{I}^i$$

Remark The introduction rules do not give us a way to distinguish the directionality of the slashes.

8.3. Alternative Notation (Sequents)

Let A, B, C stand for logic formulae (e.g. np, $np\backslash s$, $(np\backslash s)\backslash(np\backslash s)\ldots$) i.e. the categories of CG

Let Γ, Σ, Δ stand for structures (built recursively from the logical formulae by means of the $\circ$ connective) –e.g. $np \circ np\backslash s$ is a structure. $\texttt{STRUCT} := \texttt{CAT}, \texttt{STRUCT} \circ \texttt{STRUCT}$

$\Sigma \vdash A$ means that (the logic formula) A derives from (the structure) Σ (e.g. $np \circ np\backslash s \vdash s$).

$$A \vdash A$$

$$\frac{\Delta \vdash B/A \quad \Gamma \vdash A}{\Delta \circ \Gamma \vdash B}\ (/\text{E}) \qquad \frac{\Gamma \vdash A \quad \Delta \vdash A\backslash B}{\Gamma \circ \Delta \vdash B}\ (\backslash\text{E})$$

$$\frac{\Delta \circ A \vdash B}{\Delta \vdash B/A}\ (/\text{I}) \qquad \frac{A \circ \Delta \vdash B}{\Delta \vdash A\backslash B}\ (\backslash\text{I})$$

9. Lambek calculus. Elimination rule

$$\frac{np \vdash np \quad np\backslash s \vdash np\backslash s}{\underbrace{np}_{\text{sara}} \circ \underbrace{np\backslash s}_{\text{walks}} \vdash s}$$

$$\frac{np \vdash np \qquad \dfrac{(np\backslash s)/np \vdash (np\backslash s)/np \quad np \vdash np}{(np\backslash s)/np \circ np \vdash np\backslash s}}{\underbrace{np}_{\text{sara}} \circ (\underbrace{(np\backslash s)/np}_{\text{knows}} \circ \underbrace{np}_{\text{mary}}) \vdash s}$$

9.1. Lambek calculus. Subject relative pronoun

$$\underbrace{\text{The student who } [[\ldots] \text{ knows Mary}]_{s}}_{np} \underbrace{\text{left}}_{np\backslash s}$$

$$\frac{(n\backslash n)/(np\backslash s) \vdash (n\backslash n)/(np\backslash s) \qquad \dfrac{(np\backslash s)/np \vdash (np\backslash s)/np \quad np \vdash np}{(np\backslash s)/np \circ np \vdash np\backslash s}}{\underbrace{(n\backslash n)/(np\backslash s)}_{\text{who}} \circ (\underbrace{(np\backslash s)/np}_{\text{knows}} \circ \underbrace{np}_{\text{mary}}) \vdash n\backslash n}$$

Exercise: Try to do the same for relative pronoun in object position. e.g. the student who Mary met (i.e. prove that it is of category np. Which should be the category for a relative pronoun (e.g. who) that places the role of an object?

10. Lambek calculus. Introduction rule

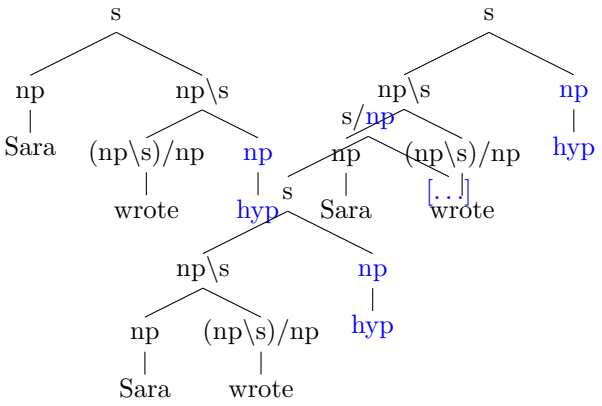
Note, below for simplicity, I abbreviate structures with the corresponding linguistic structures.

$\underbrace{\text{The book which [Sara wrote [...]]}_s}_{np}\ \underbrace{\text{is interesting}}_{np\backslash s}$.

$$\dfrac{\text{which} \vdash (n\backslash n)/(s/np) \qquad \dfrac{\text{Sara} \vdash np \qquad \dfrac{\text{wrote} \vdash (np\backslash s)/np \quad [np \vdash np]^1}{\text{wrote } np \vdash np\backslash s}\,(/\text{E})}{\text{Sara wrote } np \vdash s}\,(\backslash\text{E})}{\text{Sara wrote} \vdash s/np}\,(/\text{I})^1}{\text{which Sara wrote} \vdash n\backslash n}\,(/\text{E})$$

Introduction rules accounted for extraction.

11. Extraction: Right-branch (tree)



12. Structural Rules

Notice, to handle discontinuity phenomena we need to make use of structural rewriting. For instance, "which Sara wrote [...]" requires (some form of) associativity. "which" $\in (n\backslash n)/(s/np)$

$$\dfrac{(n\backslash n)/(s/np) \vdash (n\backslash n)/(s/np) \qquad \dfrac{\dfrac{np \vdash np \qquad \dfrac{(np\backslash s)/np \vdash (np\backslash s)/np \quad [np \vdash np]^1}{(np\backslash s)/np \circ np \vdash np\backslash s}\,(\backslash\mathrm{E})}{(np \circ ((np\backslash s)/np \circ np)) \vdash s}\,(\mathrm{Ass})}{(np \circ (np\backslash s)/np) \circ np \vdash s}\,(/\mathrm{I})^1}{np \circ (np\backslash s)/np \vdash s/np}\,(/\mathrm{E})}{\underbrace{(n\backslash n)/(s/np)}_{\text{which}} \circ (\underbrace{np}_{\text{sara}} \circ \underbrace{(np\backslash s)/np}_{\text{wrote}}) \vdash n\backslash n}\,(/\mathrm{E})$$

12.1. Structural Rules: Formally (Advanced!)

Structural rules are rule governing the structure we built while applying logical rules. Associativity and Permutativity (or Commutativity) are example of structural rules.

Starting from the a Logic that consists only of the Logical rules we have seen we can define a **family of Logics** that differ on their structural properties.

Hence we speak of the Lambek Calculi. The base one consists only of logical rules (NL).

(Side Remark: Structural rules correspond to model theoretical properties.)

Structural rules. Let us write $\Gamma[\Delta]$ for a structure Γ contaning a distinguished occurrence of the substructure Δ. Adding a structural rule of Associativity [ass] to NL, one obtains L. By adding commutativity [per] to L one obtains LP, and so on.

For instance,

$$\frac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash C}\ (\text{ass}) \qquad \frac{\Gamma[(\Delta_2 \circ \Delta_1)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2)] \vdash C}\ (\text{per})$$

12.2. Structural Rules and NL

But

- global structural rules are "unsound" when reasoning with natural language.

  I.e. The logical grammar will overgenerate proving as grammatical also ungrammatical sentence.

(Local) Structural Rules have been used to account for cross-linguistics variations.
(be happy if you get the intuitive idea)

13. Historical Introduction: Syntax-Semantic Interface

- **Who**: van Benthem (1987), Buszkowski (1987)
- **Aim**: Syntax-Semantic interface
- **How**: Curry-Howard Correspondence between proofs and terms.

$$x : A \vdash x : A$$

$$\frac{\Gamma \vdash t : A/B \quad \Delta \vdash u : B}{\Gamma \circ \Delta \vdash t(u) : A}\ (/\mathrm{E}) \qquad \frac{(\Gamma \circ x : B) \vdash t : A}{\Gamma \vdash \lambda x.t : A/B}\ (/\mathrm{I})$$

$$\frac{\Delta \vdash u : B \quad \Gamma \vdash t : B\backslash A}{\Delta \circ \Gamma \vdash t(u) : A}\ (\backslash\mathrm{E}) \qquad \frac{(x : B \circ \Gamma) \vdash t : A}{\Gamma \vdash \lambda x.t : B\backslash A}\ (\backslash\mathrm{I})$$

13.1. Semantics: Examples

The book which Sara wrote

$$\dfrac{\dfrac{\text{sara} \vdash np : \texttt{sara} \qquad \dfrac{\text{wrote} \vdash (np\backslash s)/np : \texttt{wrote} \quad [z \vdash np : z]^1}{\text{wrote } z \vdash np\backslash s : \texttt{wrote}(z)}\,(/\text{E})}{\text{sara wrote } z \vdash s : \texttt{wrote}(z)(\texttt{sara})}\,(\backslash\text{E})}{\text{sara wrote} \vdash s/np : \lambda z.\texttt{wrote}(z)(\texttt{sara})}\,(/\text{I})^1$$

$$\Downarrow$$

The introduction rules correspond to λ-abstraction.

13.2. NP and quantified NP

John and one student left.

We can assign to John the category np and term assignment `john` and **derive** the category and term of quantified np.

$$\dfrac{\dfrac{\text{john} \vdash np : \texttt{john} \quad [P \vdash np\backslash s : P]^1}{\text{john } P \vdash s : P(\texttt{john})}\,(\backslash\text{E})}{\text{john} \vdash s/(np\backslash s) : \lambda P.P(\texttt{john})}\,(/\text{I})^1$$

We have proved: $np \vdash s/(np\backslash s)$. This means, we can assign John the category np (considering it an entity, i.e. a term of type e) and derive from it the **higher order category** of quantified NP as it would be necessary for, e.g. coordination of a NP and a QP.

Exercise What about "Mary saw John and one student"?

13.3. Remarks

First of all, note how the system assigns a variable to the hypothesis. The latter is **discharged** by means of [/I] (or [\I]) which corresponds to the abstraction over the variable.

Moreover, note that the higher order types in the derivation I gave and the one you have found with the exercise are different, but they correspond to the **same lambda terms**, i.e. the two structures are correctly assigned the same meaning.

Starting from the labelled lexicon, the task for the Lambek derivational engine is to compute the lambda term representing the meaning assembly for a complex structure as a **by-product** of the derivation that establishes its grammaticality.

14. CTL. Derivational vs. Lexical Meaning

Meaning representation can be computed in two ways.

- **Lexical** one starts labeling the axioms of a derivation with the actual lambda terms assigned in the lexicon.
- **Derivational** one labels the leaves of the derivation with variables, computes the proof term for the final structure and then replaces the variables by the actual lambda terms assigned in the lexicon to the basic constituents.

14.1. Example: Relative Clause

The relative clause examples offer a nice illustration of the division of labor between lexical and derivational semantics.

Intuitively, a relative pronoun has to compute the **intersection of two properties**: the common noun property obtained from the n that is modified, and the property obtained from the body of the relative clause, a sentence with a np hypothesis missing.

In the logical form, this would come down to **binding two occurrences** of a variable by one λ binder.

On the level of **derivational** semantics, one **cannot obtain this double binding**: the Lambek systems are resource sensitive, which means that every assumption is used exactly once. (see later Section 15)

14.2. Relative Clause: Double binding

But on the level of **lexical** semantics, we can overcome this expressive limitation (which is syntactically well-justified!) by assigning the relative pronoun a double-bind term as its lexical meaning recipe:

$$\text{which} \in (n\backslash n)/(s/np) : \lambda X.\lambda Y.\lambda z.X(z) \wedge Y(z).$$

In this way, we obtain the proper recipe for the relative clause **which Sara wrote**, namely $\lambda Y.\lambda z.\texttt{wrote}(\texttt{Sara}, z) \wedge Y(z)$.

Exercise Build the meaning representation of "which sara wrote" by applying labelled logical rules.

14.3. Relative Clause: derivational meaning

$$\dfrac{\text{which} \vdash (n\backslash n)/(s/np) : X_4 \qquad \dfrac{\dfrac{\text{Sara} \vdash np : X_3 \qquad \dfrac{\text{wrote} \vdash (np\backslash s)/np : X_1 \quad [x \vdash np : X_2]^1}{\text{wrote} \circ x \vdash np\backslash s : X_1X_2}\,(/\text{E})}{\text{Sara} \circ (\text{wrote} \circ x) \vdash s : (X_1X_2)X_3}\,(\backslash\text{E})}{(\text{Sara} \circ \text{wrote}) \circ x \vdash s : (X_1X_2)X_3}\,(\text{ass})}{\text{Sara} \circ \text{wrote} \vdash s/np : \lambda X_3.(X_1X_2)X_3}\,(/\text{I})^1}{\text{which} \circ (\text{Sara} \circ \text{wrote}) \vdash n\backslash n : X_4(\lambda X_3.(X_1X_2)X_3)}\,(/\text{E})$$

By replacing the variables $X_1, \ldots, X_4$ with the corresponding lexical assignments, and applying the reduction rules, one obtains the proper meaning of the analyzed structure.

Note, the structural rules do not effect the meaning assembly.

15. Fragment of Lambda Terms (Advanced!)

The Lambek calculi are **fragments of intuitionistic implicational logic**.

Consequently, the lambda terms computed by it form a **fragment of the full language of lambda terms**.

First of all, since empty antecedents are not allowed and the Lambek calculi are resource sensitive, viz. each assumption is used exactly once, the system reasons about lambda terms with specific properties:

(i) each subterm contains a free variable; and

(ii) no multiple occurrences of the same variable are present. The latter could seem to be too strong constraint when thinking of linguistic applications. However, this is not the case as we have discuss above by looking at the relative pronoun.

(iii) each occurrence of the λ abstractor in $\alpha \in \mathsf{TERM}$ binds a variable within its scope. (resource sensitive!)

15.1. Curry-Howard Correspondence (Advanced!)

Derivations for the various Lambek calculi are all associated with LP (the associative and permutative Lambek Calculus) term recipes. Therefore, we move from an isomorphism to a weaker **correspondence**.

Theorem 15.1 Given an LP derivation of a sequent $A_1, \ldots, A_n \vdash B$ one can find a corresponding construction $\alpha_a \in \Lambda(\mathsf{LP})$, and conversely. A term $\alpha_a \in \Lambda(\mathsf{LP})$ is called a construction of a sequent $A_1, \ldots, A_n \vdash B$ iff α has exactly the free variable occurrences $x^1_{\mathtt{type}(A_n)}, \ldots, x^n_{\mathtt{type}(A_n)}$.

15.2. Normal Form (Advanced!)

An important notion of the lambda calculi is "normal form" terms that are obtained proof theoretically by defining normal form derivations as following.

Definition 15.2 (Normal Form for Natural Deduction Derivations)) A derivation in natural deduction format is in **normal form** when there are no detours in it. A **detour** is formed when

i. a connective is introduced and immediately eliminated at the next step.
ii. an elimination rule is immediately followed by the introduction of the same connective.

The rules eliminating these two detours are called **reduction** rules.

Remark The reductions of the detours in *i.* and in *ii.* correspond to β-reduction and η-reduction, respectively. Moreover, note that the above rewriting rules hold for all Lambek calculi, regardless of their structural rules.

15.3. Normal form proof: example (Advanced!)

By means of example, we give the reduction rule corresponding to η-reduction.

$$\dfrac{\dfrac{\begin{array}{c}D_1\\ \vdots\\ [B \vdash x : B]^1 \quad \Gamma \vdash t : B\backslash A\end{array}}{B \circ \Gamma \vdash t(x) : A}\,(\backslash\mathrm{E})}{\Gamma \vdash \lambda x.t(x) : B\backslash A}\,(\backslash\mathrm{I})^1 \qquad \text{rewrites to} \qquad \begin{array}{c}D_1\\ \vdots\\ \Gamma \vdash t : B\backslash A\end{array}$$

in the lambda-calculus the reduction above corresponds to the rewrite rule $\lambda x.t(x) \Rightarrow_\eta t$

The correspondence between proofs and lambda terms is completed by the following theorem.

Theorem 15.3 (Normalization) If $\mathcal{D}$ is a normal form derivation of $x_1 : A_1, \ldots x_n : A_n \vdash \alpha : C$, then α is in β, η normal form.

16. From CG to NL

- Classical Categorial Grammar consists of (only) function application rules. But,
- Concatenative function application is not enough to analyze natural language.
- We need to compose as well as decompose structures.

By moving from a rule-based approach to a logical system we obtain abstraction, (\I) and (/I) besides function application, (\E) and (/E). Hence, we obtain

1. **derivability relations** among types
2. a way to **decompose** built structures

From CG to NL,

CG	NL
Categories	Formulas
Category forming operators	Logical Operators
Rule schemata	Inference Rules
Parsing	Deduction

17. Lambek calculus. Advantages

- **Hypothetical reasoning:** Having added [\I], [/I] gives the system the right expressiveness to reason about hypothesis and abstract over them.
- **Curry Howard Correspondence:** Curry-Howard correspondence holds between proofs and terms. This means that parsed structures are assigned an interpretation into a model via the connection 'categories-terms'.
- **Logic:** We have moved from a grammar to a logic. Hence its behavior can be studied. The system is sound, complete and decidable.

18. Residuation

Interestingly enough, the operators of the Lambek calculi are rather well known operators. They behave like basic operation of Maths, like $:, x$. All pair of operators sharing this property are known as "residuated operators".

$$x \times y \leq z \text{ iff } x \leq \frac{z}{y}$$

$$\frac{x \times y \leq z}{x \leq \frac{z}{y}} \qquad \frac{A \circ B \vdash C}{A \vdash C/B}$$

Based on this observation (pointed out by Lambek), Michael Moortgat and Natasha Kurtonina, extended the language of the Lambek Calculi to **unary operators** characterized by the same property, namely $\Diamond$ and $\blacksquare$ which turned out to be already known in Logic (see Temporal Logic) and to be able to model the feature checking mechanism that we have seen is required when analyzing NL, e.g. feature agreement.

19. Summing up

The main points of today topic to be kept in mind are the following:

1. Linguistic signs are **pairs of form and meaning**, and composed phrases are structures rather than strings.
2. When employing a logic to model linguistic phenomena, **grammatical derivations are seen as theorems** of the grammatical logic.
3. The correspondence between proofs and natural language models, via the lambda terms, properly accounts for the natural language **syntax semantics interface**.

Reference on CG and Lambek Calculi: First chapter of my thesis.

20. What have we learned?

- We've seen we can exploit **derivability relations** to control composition of types. (e.g. NP coor QP)
- However, we have not found yet the type for the relative pronoun that grasps its behavior and its link with the dependent object, properly. For instance, if we modify the context slightly

  "which Sara wrote there" cannot be recognized by NL with the type assigned to "which".

- We should still understand how to properly use
  - **structural rules**
  - **derivability** relations,
  - **unary operators** logical rules,
  - how to **lexically control** their application.

Project Study the four points above and build a fragment of CTL covering long-distance phenomena examples. Literature: Moortgat 02.