1. 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**.

1.1. 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 1.1 (Types) Given a non-empty set of basic types Base, the set of types TYPE is the smallest set such that

i. Base $\subseteq$ TYPE;
ii. $(a \rightarrow b) \in$ TYPE, if a and $b \in$ 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 $(a \rightarrow b)$. A function mapping the syntactic categories into TYPE can be given as follows.

Definition 1.2 (Categories and Types) *Let us define a function* $\mathtt{type} : \mathit{CAT} \rightarrow \mathit{TYPE}$ *which maps syntactic categories to semantic types.*

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

1.2. 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

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

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

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

2. Compositionally vs. Non-compositionally

An alternative approach:

- In **compositional** semantics theory the relation between the meaning of an expression and the meaning of its constituents is a **function**: to each distinct syntactic structure correspond a distinct interpretation.
- In **underspecification** theory this relation is systematic but it's **not a function**: an expression analyzed by a single syntactic structure can be associated with a set of alternative interpretations rather than with a unique semantic value. Sentences are assigned underspecified representation containing parameters whose value can be defined in several distinct ways. Constraints apply to filter the possible combinations of values for the set of parameters in such a schematic representation.

Reference: For underspecified semantics see BB1.

3. Conclusions I

While working with the lambda-terms we have seen we need **abstraction** to, e.g. to account for the different ways quantified NP can scope.

But in Categorial Grammar there is no way to abstract from a built structure.

Now we will see the missing ingredient (abstraction at syntactic level) allows us to move from a formal grammar to a logic (a **logical grammar**).

We will look at Lambek Calculi and their application to NL.

4. 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 know them, it 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.

4.1. 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:

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

4.2. 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.

4.3. 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. $\texttt{STRUCT} := \mathsf{CAT}, \texttt{STRUCT} \circ \texttt{STRUCT}$

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

$$A \vdash A$$

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

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

5. Lambek calculus. Elimination rule

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

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

5.1. 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 plays the role of an object?

6. Lambek calculus. Introduction rule

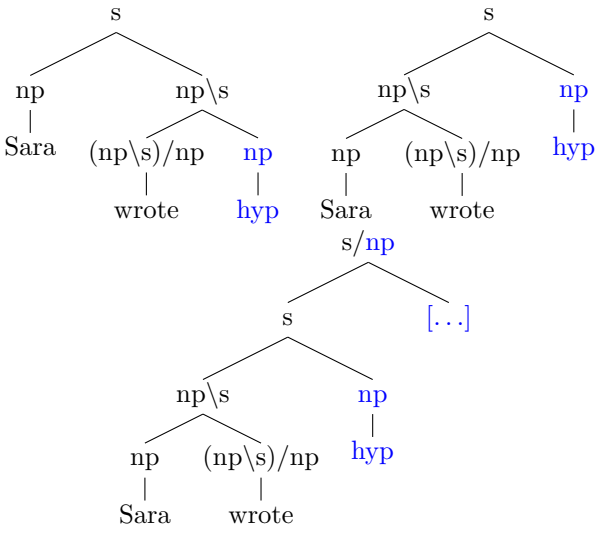
Note, below for simplicity, I abbreviate structures with the corresponding linguistic structures.

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

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

Introduction rules accounted for extraction.

7. Extraction: Right-branch (tree)



8. 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" $\in (n\backslash n)/(s/np)$

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

8.1. 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,

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

8.2. 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)

9. Historical Introduction: Syn.-Sem. Interface

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

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

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

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

9.1. Semantics: Examples

The book which Sara wrote

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

$$\Downarrow$$

The introduction rules correspond to λ-abstraction.

9.2. NP and quantified NP

John and one student left.

We can assign to John the category np and term assignment $\texttt{john}$ 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"?

9.3. 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.

10. 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

11. 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.

12. Summing up

The main points of the second part of today lesson 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.

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

14. Next Time

- In the lab we will practice with CG and lambda terms, and with Lambek calculus and lambda terms.
- December 2: In the lecture we will compare Formal Grammars, and go back to the questions "Is Natural Language Context Free?"
- December 9: We will look at parsing.
- December 16: Sample exam (2 hrs)