Computational Linguistics: Syntax-Semantics

Raffaella Bernardi

KRDB, Free University of Bozen-Bolzano

Via della Mostra 4, Room: 1.06, e-mail:

bernardi@inf.unibz.it

Contents

1. Recall: Formal Semantics Main questions

The main questions are:

1. What does a given sentence mean?
2. How is its meaning built?
3. How do we infer some piece of information out of another?

1.1. Generalized Quantifiers

$$
\begin{array}{lcl}
[\![\texttt{no man}]\!] & = & \{X \subseteq E \mid [\![\texttt{man}]\!] \cap X = \emptyset\}. \\
[\![\texttt{some man}]\!] & = & \{X \subseteq E \mid [\![\texttt{man}]\!] \cap X \neq \emptyset\}. \\
[\![\texttt{every man}]\!] & = & \{X \subseteq E \mid [\![\texttt{man}]\!] \subseteq X\}. \\
[\![\texttt{man which VP}]\!] & = & [\![\texttt{man}]\!] \cap [\![\texttt{VP}]\!].
\end{array}
$$

Therefore, determiners are as below:

$$
\begin{array}{lcl}
[\![\texttt{no N}]\!] & = & \{X \subseteq E \mid [\![\texttt{N}]\!] \cap X = \emptyset\}. \\
[\![\texttt{some N}]\!] & = & \{X \subseteq E \mid [\![\texttt{N}]\!] \cap X \neq \emptyset\}. \\
[\![\texttt{every N}]\!] & = & \{X \subseteq E \mid [\![\texttt{N}]\!] \subseteq X\}. \\
[\![\texttt{N which VP}]\!] & = & [\![\texttt{N}]\!] \cap [\![\texttt{VP}]\!].
\end{array}
$$

Generalized quantifiers have attracted the attention of many researchers working on the interaction between logic and linguistics.

2. Determiners

Which is the lambda term representing quantifiers like “nobody”, “everybody”, “a man”or “every student” or a determiners like “a”, “every” or “no” ?

the term representing “a” is:

$$\lambda X.\lambda Y.\exists z.X(z) \wedge Y(z)$$

Try to obtain the meaning representation for “a man”, and the “a man loves Mary”.

By β-conversion twice we obtain that “a man” is $\lambda Y.\exists z.\texttt{Man}(z) \wedge Y(z)$, and then $\exists z.\texttt{Man}(z) \wedge love(z, mary)$

3. Ambiguities

How many meanings has the sentence "John didn't read a book."?

Starting from:

john: $\mathtt{j}$	book: $\lambda x(\mathtt{book}(x))$
read: $\lambda x.\lambda y.(\mathtt{read}(y,x))$	didn't: $\lambda X.\lambda y.\neg X(y)$
a: $\lambda X.\lambda Y(\exists x.X(x) \wedge Y(x))$	

build the meaning representation for "John didn't read a book".

a. $\exists x.\mathtt{book}(x) \wedge \neg\mathtt{read}(\mathtt{j},x)$ [A > NOT]

b. $\neg\exists x.\mathtt{B}(x) \wedge \mathtt{read}(\mathtt{j},x)$ [NOT > A]

- **Scope**: In a. the quantifier phrase (QP), "a book", has scope over "didn't" [A > NOT], whereas in b. it has narrow scope [NOT > A].
- **Binding**: the variable x is bound by "a book" in "John didn't read a book".

3.1. Scope Ambiguities

Can you think of other expressions that may cause scope ambiguity?

John **think** a student left

Does the student exist or not?

a. $\exists x.think(j, left(x))$

b. $think(j, \exists x.left(x))$

4. Dependencies

While studying the syntax of natural language, we have seen that important concepts to account for are local and long-distance dependencies.

The λ-operator gives us (more or less) a way to represent this link semantically.

For instance, in $\lambda x.\lambda y.like(y, x)$ we express that the dependency of the subject and object from the verb.

But the calculus gives us also a natural way to handle long-distance dependencies: eg. relative pronouns.

4.1. Relative Pronouns

For instance, "which John read [...]":

We know how to represent the noun phrase "John" and the verb "read", namely, as `john` and $\lambda x.y.\texttt{read}(y, x)$.

What is the role of "which" in e.g. "the book which John read is interesting"?

The term representing "which" has to express the fact that it is replacing the role of a noun phrase in subject (or object position) within a subordinate sentence while being the subject (object) of the main sentence:

$$\lambda X.\lambda Y.\lambda z.X(z) \wedge Y(z)$$

The double role of "which" is expressed by the double occurrence of z.

4.2. Relative Pronoun (Cont'd)

Recall,

$$\lambda X.\lambda Y.\lambda z.X(z) \wedge Y(z)$$

i. read u: $\lambda y(\texttt{read}(y, u)$
ii. John read u: $\texttt{read}(\texttt{j}, u)$
iii. John read: $\lambda u.\texttt{read}(\texttt{j}, u)$
iv. which John read: $\lambda Y.\lambda z.\texttt{read}(\texttt{j}, z) \wedge Y(z)$

- at the syntactic level we said that the relative pronoun "which" plays the role of the verb's object and it leaves a gap in the object position.
- Semantically, the gap is represented by the u on which the relative pronoun forces the abstraction [iii.] before taking its place.

5. Summing up: Constituents and Assembly

Let's go back to the points where FOL fails, i.e. constituent representation and assembly. The λ-calculus succeeds in both:

Constituents: each constituent is represented by a lambda term.

John: $\mathtt{j}$ knows: $\lambda xy.(\mathtt{know}(x))(y)$ read john: $\lambda y.\mathtt{know}(y, \mathtt{j})$

Assembly: function application $(\alpha(\beta))$ and abstraction $(\lambda x.\alpha[x])$ capture composition and decomposition of meaning representations.

6. The Syntax-Semantics Interface

So far, we have spoken of syntax and semantics of natural language as two distinct and separate levels. However, as we know from our every-day use of NL these levels are tiedely connected.

Today, we will look at the interface between syntax and semantic.

Recall, from syntax we know that phrases are composed out of words, and from semantics we know that **meaning flows from the lexicon**.

Reference : L.T.F. Gamut "Logic, Language and Meaning", Vol. 2. The University of Chicago Press,1991. Chapter 4. (see library or ask copies to me)

Moortgat's talk : Montague's approach: Compositionality as a homomorphism syntax $\rightsquigarrow$ semantics. Curry-Howard: Formulas-as-types/proofs-as-programs.

6.1. Augmenting 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.

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

6.2. Montague Universal Grammar

The rule-to-rule and lambda techniques are used in the approach to natural language semantics developed by Richard Montague. In his theory, there are

- **syntactic rules** which show how constituents maybe combined to form other constituents.
- **translation rules** (associated with each such syntax rule) which show how the logical expressions for the constituents have to be joined together to form the logical form of the whole.

For instance, the syntactic and semantics rule for composing and NP with and IV:

S2: If $\delta \in P_{IV}$ and $\alpha \in P_{NP}$, then $F_1(\alpha, \delta) \in P_S$ and $F_1(\alpha, \delta) = \alpha\delta'$, where δ' is the result of replacing the main verb in δ by its third-person singular present form.

T2: If $\delta \in P_{IV}$ and $\alpha \in P_{NP}$ and $\delta | \rightarrow \delta'$ and $\alpha | \rightarrow \alpha'$, then $F_1(\alpha, \delta) | \rightarrow \alpha'(\delta')$.

As grammar, he used Categorial Grammar.

7. 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:

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

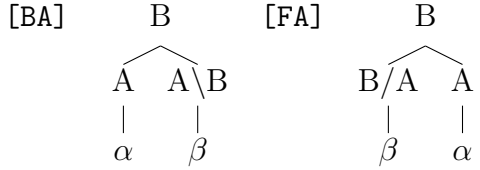
8. CG: Syntactic Rules

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.



9. CG Lexicon: Toy Fragment

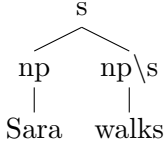
Let ATOM be $\{n, s, np\}$ (for nouns, sentences and noun phrases, respectively) and LEX as given below. Recall PSG rules: $np \to det\ n, s \to np\ vp, vp \to v\ np \ldots$

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]



10. Classical Categorial Grammar

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

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

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

11. 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

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

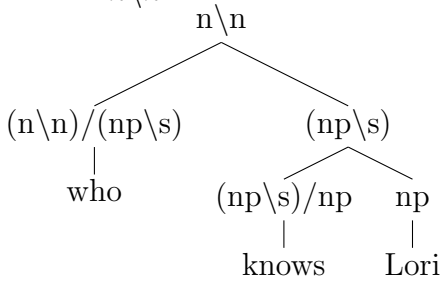
11.1. Relative Pronoun

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

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



11.2. 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

12. 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**.

12.1. 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 12.1 (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 12.2 (Categories and Types) *Let us define a function* $\mathtt{type} : \mathit{CAT} \rightarrow \mathit{TYPE}$ *which maps syntactic categories to semantic types.*

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

12.2. 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

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

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

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

13. 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. Possible topic for a project.

14. Next Time

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.

Next week 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.