Computational Linguistics: Semantics II

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. Building Meaning Representations

To build a meaning representation we need to fulfill three tasks:

Task 1 Specify a reasonable **syntax** for the natural language fragment of interest.

Task 2 Specify semantic representations for the **lexical items**.

Task 3 Specify the **translation** of constituents **compositionally**. That is, we need to specify the translation of such expressions in terms of the translation of their parts, parts here referring to the substructure given to us by the syntax.

Moreover, when interested in Computational Semantics, all three tasks need to be carried out in a way that leads to computational implementation naturally.

1.2. Lambda-calculus: Functional Application

Summing up:

- FA has the form: Functor(Argument). E.g. $(\lambda x.love(x, mary))(john)$
- FA triggers a very simple operation: Replace the λ-bound variable by the argument. E.g. $(\lambda x.love(x, mary))(john) \Rightarrow love(john, mary)$

Exercise 1 and 2.

2. Extending the lexicon

Before we have left open the question of what does an expression like "a" contribute to? FOL does not give us the possibility to express it's meaning representation. We will see now that instead lambda terms provide us with the proper expressivity.

2.1. Quantified NP

a) Every South African student of the EM in CL attends the Comp Ling course.

b) No South African student of the EM in CL attend the Logic course.

a) means that if Thomas and Ronell constitute the set of the South African students of the EM in CL, then it is true for both of them that they attend the Comp. Ling course.

b) means that for none of the individual members of the set of South African students of the EM in CL it is true that he attends the Logic course.

What is the interpretation of "every South African student" and of "no South African student"?

Individual constants used to denote specific individuals cannot be used to denote quantified expressions like "every man", "no student", "some friends".

Quantified-NPs like "every man", "no student", "some friends" are called non-referential.

2.2. Generalized Quantifiers

A Generalized Quantifier (GQ) is a set of properties, i.e. **a set of sets-of-individuals**.

For instance, "every man" denotes the set of properties that every man has. The property of "walking" is in this set iff every man walks. For instance,

$$
\begin{array}{lcl}
[\![\texttt{man}]\!] & = & \{a, b, c\}; \\
[\![\texttt{fat}]\!] & = & \{a, b, c, d\}; \\
[\![\texttt{dog}]\!] & = & \{d\}; \\
[\![\texttt{run}]\!] & = & \{a, b\}; \\
[\![\texttt{jump}]\!] & = & \{b, c, d\}; \\
[\![\texttt{laugh}]\!] & = & \{b, d\};
\end{array}
$$

Which is the interpretation of "every man"?

$$[\![\texttt{every man}]\!] = \{X | [\![\texttt{man}]\!] \subseteq X\} = \{\{a, b, c\}, \{a, b, c, d\}\} = \{[\![\texttt{man}]\!], [\![\texttt{fat}]\!]\}$$

2.3. 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.

3. Determiners

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

We know how to represent in FOL the following sentences

- “Nobody left”
  $\neg\exists x.\mathtt{left}(x)$
- “Everybody left”
  $\forall x.\mathtt{left}(x)$
- “Every student left”
  $\forall x.\mathtt{Student}(x) \rightarrow \mathtt{left}(x)$
- “A student left”
  $\exists x.\mathtt{Student}(x) \wedge \mathtt{left}(x)$
- “No student left”
  $\neg\exists x.\mathtt{Student}(x) \wedge \mathtt{left}(x)$

But how do we reach these meaning representations starting from the lexicon?

3.1. Determiners (cont'd)

Let's start representing "a man" as $\exists x.man(x)$. Applying the rules we have seen so far, we obtain that the representation of "A man loves Mary" is:

$$love(\exists x.man(x), mary)$$

which is clearly wrong.

Notice that $\exists x.man(x)$ just isn't the meaning of "a man". If anything, it translates the complete sentence "There is a man".

3.2. Determiners (Cont'd)

Let's start from what we have, namely "man" and "loves Mary":

$\lambda y.man(y)$, $\lambda x.love(x, mary)$.

Hence, 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 $\exists z.\texttt{Man}(z) \wedge love(z, mary)$

4. Ambiguities

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

Starting from:

john: $\texttt{j}$	book: $\lambda x(\texttt{book}(x))$
read: $\lambda x.\lambda y.(\texttt{read}(y, x))$	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. $\exists x.\texttt{book}(x) \wedge \neg\texttt{read}(\texttt{j}, x)$ [A > NOT]

b. $\neg\exists x.\texttt{B}(x) \wedge \texttt{read}(\texttt{j}, x)$ [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".

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

5. 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.

5.1. Relative Pronouns

For instance, “which John read [...]”:

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

What is the role of “which” in e.g. “the book which John read is read”?

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.

5.2. 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.

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

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