Computational Linguistics: Semantics

Raffaella Bernardi

KRDB, Free University of Bozen-Bolzano

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

Contents

1. CL: what have we achieved?

Main challenge of CL: to deal with ambiguity at the different levels.

Which kind of ambiguities are the following ones?

- La vecchia porta sbatte.
- La vecchia porta la sbarra.
- John saw the man with the telescope.
- John saw the woman in the park with the telescope. He was at home.

We have seen how to recognize/parse these sentences so to obtain different parse trees whenever necessary.

2. Semantics

Semantics: it's the study of the meaning of words and the reference of linguistic inputs. The former is studied in **Lexical Semantics** and the latter in **Formal Semantics**.

Lexical Semantics Words are seen as having a systematic structure that governs what they mean, how they **relate to actual entities** and how they can be used. Studies on this topic result into e.g. Dictionary or Ontologies like WordNET.

whereas we will look at Formal Semantics.

3. 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?

The first and last question are closely connected.

In fact, since we are ultimately interested in understanding, explaining and accounting for the entailment relation holding among sentences, we can think of **the meaning of a sentence as its truth value**.

4. Logical Approach

To tackle these questions we will use Logic, since using Logic helps us answering the above questions at once.

1. Logics have a precise semantics in terms of **models** —so if we can translate/represent a natural language sentence S into a logical formula ϕ, then we have a precise grasp on at least part of the meaning of S.
2. Important **inference problems** have been studied for the best known logics, and often good **computational implementations** exists. So translating into a logic gives us a handle on inference.

When we look at these problems from a computational perspective, i.e. we bring in the implementation aspect too, we move from Formal Semantics to **Computational Semantics**.

4.1. Model: individual constants

The interpretation of a formal language has to include a specification of what constants in the language refers to. This is done by means of the **interpretation function** $\mathcal{I}$ which assigns an appropriate **denotation** in the model $\mathcal{M}$ to each individual and n-place predicate constant .

If α is an individual constant, $\mathcal{I}$ maps α onto one of the entities of the universe of discourse $\mathcal{U}$ of the model $\mathcal{M} : \mathcal{I}(\alpha) \in \mathcal{U}$.

1 `picture: universe of discourse set U with dots.`

4.2. Model:one-place predicates

One-place properties are seen as sets of individuals: the property of being orange describes the **set of individuals** that are orange. Formally, for P a one-place predicate, the interpretation function $\mathcal{I}$ maps P onto a subset of the universe of discourse $\mathcal{U}: \mathcal{I}(P) \subseteq \mathcal{U}$. For instance,

2 `picture: universe with a subset standing for orange`

4.3. Model: two-place predicates

Two-place predicates such as “love”, “eat”, “mother-of” do not denote sets of individuals, but **sets of ordered pairs of individuals**, namely all those pairs which stand in the “loving”, “eating”, “mother-of” relations. We form ordered pairs from two sets A and B by taking an element of A as first member of the pair and an element of B as the second member. Given the relation R, the interpretation function $\mathcal{I}$ maps R onto a set of ordered pairs of elements of $\mathcal{U} : \mathcal{I}(R) \subseteq \mathcal{U} \times \mathcal{U}$

3 `picture: page 78`

4.4. Example

Let our model be based on the set of entities $E = \{\texttt{lori}, \texttt{ale}, \texttt{sara}, \texttt{pim}\}$ which represent **Lori, Ale, Sara** and **Pim**, respectively. Assume that they all know themselves, plus **Ale** and **Lori** know each other, but they do not know **Sara** or **Pim**; **Sara** does know **Lori** but not **Ale** or **Pim**. The first three are students whereas **Pim** is a professor, and both **Lori** and **Pim** are tall. This is easily expressed set theoretically. Let $[\![\texttt{w}]\!]$ indicate the interpretation of **w**:

$$\begin{array}{lcl}
[\![\texttt{sara}]\!] & = & \text{sara;} \\
[\![\texttt{pim}]\!] & = & \text{pim;} \\
[\![\texttt{lori}]\!] & = & \text{lori;} \\
[\![\texttt{know}]\!] & = & \{\langle \text{lori, ale} \rangle, \langle \text{ale,lori} \rangle, \langle \text{sara, lori} \rangle, \\
 & & \langle \text{lori, lori} \rangle, \langle \text{ale, ale} \rangle, \langle \text{sara, sara} \rangle, \langle \text{pim, pim} \rangle\}; \\
[\![\texttt{student}]\!] & = & \{\text{lori, ale, sara}\}; \\
[\![\texttt{professor}]\!] & = & \{\text{pim}\}; \\
[\![\texttt{tall}]\!] & = & \{\text{lori, pim}\}.
\end{array}$$

which is nothing else to say that, for example, the relation **know** is the **set of pairs** $\langle \alpha, \beta \rangle$ where α knows β; or that ‘student’ is the set of all those elements which are a student.

4.5. Quantified NP

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

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

a) means that if Sergio and Luis constitute the set of the Mexican 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 Mexican student of the EM in CL it is true that he attends the Logic course.

What is the interpretation of "every Mexican student" and of "no Mexican 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.

4.6. 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,

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

Which is the interpretation of "every man"?

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

4.7. 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.

4.8. Relational and Functional Perspectives

Alternatively, one can assume a functional perspective and interpret, for example, **student** as a function from individual (entities) to truth values, $student(monika) = 1$, $student(raffaella) = 0$.

The shift from the relational to the functional perspective is made possible by the fact that the **sets and their characteristic functions amount to the same thing**:

> if f_X is a function from Y to $\{0, 1\}$, then $X = \{y \mid f_X(y) = 1\}$. In other words, the assertion '$y \in X$' and '$f_X(y) = 1$' are equivalent.

Therefore, the two notations $y(z)(u)$ and $y(u, z)$ are equivalent.

4.9. Exercises: Relations vs. Functions

Think of which function you can assign to the words in the model considered before and repeated here:

Sara, Pim, Lori, know, student, professor, tall, every man, every Mexican student, no Mexican student, some man.

4.10. Summing up

Summarizing, when trying to formalize natural language semantics, at least two sorts of objects are needed to start with: the set of **truth values** t, and the one of **entities** e.

Moreover, we spoke of more complex objects as well, namely functions. More specifically, we saw that the kind of functions we need are **truth-valued functions** (or boolean functions).

Furthermore, we have illustrated how one can move back and forwards between a **set/relational and a functional perspective**. The former can be more handy and intuitive when reasoning about entailment relations among expressions; the latter is more useful when looking for lexicon assignments.

References: Keenen 85.

5. Formal Semantics: What

What does a given sentence mean?

The meaning of a sentence is its truth value. Hence, this question can be rephrased in "Which is the meaning representation of a given sentence to be evaluated as true or false?"

- **Meaning Representations**: Predicate-Argument Structures are a suitable meaning representation for natural language sentences. E.g.

  the meaning representation of "Vincent loves Mia" is $\texttt{loves}(\texttt{vicent}, \texttt{mia})$

  whereas the meaning representation of "A student loves Mia" is $\exists x.\texttt{student}(x) \wedge \texttt{loves}(x, \texttt{mia})$.

- **Interpretation**: a sentence is taken to be a proposition and its meaning is the truth value of its meaning representations. E.g.

  $[\![\exists x.\texttt{student}(x) \wedge \texttt{left}(x)]\!] = 1$ iff standard FOL (First Order Logic) definitions are satisfied.

6. Formal Semantics: How

How is the meaning of a sentence built?

To answer this question, we can look back at the example of "Vincent loves Mia". We see that:

- "Vincent" contributes the constant `vincent`
- "Mia" contributes the constant `mia`
- "loves" contributes the relation symbol `loves`

This observation can bring us to conclude that the **words** making up a sentence contribute all the bits and pieces needed to build the sentence's meaning representation.

In brief, **meaning flows from the lexicon**.

6.1. Formal Semantics: How (cont'd)

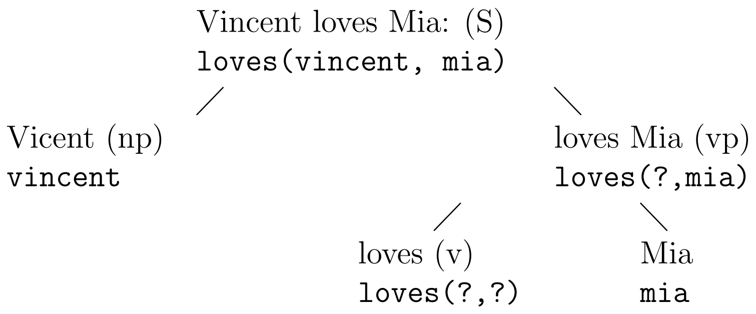
But,

1. Why the meaning representation of "Vincent loves Mia" is not `love(mia,vincent)`?
2. What does "a" contribute to in "A student loves Mia"?

As for 1., the missing ingredient is the **syntactic structure**! [Vincent [loves$_v$ Mia$_{np}$]$_{vp}$]$_s$.

We will come back to 2. in a few slides.

6.2. Formal Semantics: How (Cont'd)



Briefly, **syntactic structure guiding gluing**.

6.3. Compositionality

The question to answer is: "How can we specify in which way the bit and pieces combine?"

1. Meaning (representation) ultimately flows from the lexicon.
2. Meaning (representation) are combined by making use of syntactic information.
3. The meaning of the whole is function of the meaning of its parts, where "parts" refer to substructures given us by the syntax.

6.4. Ambiguity

A single linguistic sentence can legitimately have different meaning representations assigned to it.

For instance, "John saw a man with the telescope"

a.	John [saw [a man [with the telescope]$_{pp}$]$_{np}$]$_{vp}$	$\exists x.\texttt{Man}(x) \wedge \texttt{Saw}(j,x) \wedge \texttt{Has}(x,t)$
b.	John [[saw [a man]$_{np}$]$_{vp}$ [with the telescope]$_{pp}$]$_{vp}$	$\exists x.\texttt{Man}(x) \wedge \texttt{Saw}(j,x) \wedge \texttt{Has}(j,t)$

Different parse trees result into different meaning representations!

7. How far can we go with FOL?

FOL can capture the **what** (partially) and cannot capture the **how**, i.e.

Problems with the "what":

order
- Swimming is healthy. $Healthy(Swim)$: wrong! (property of property)
- John has all the properties of Santa Clause $\forall P(P(s) \rightarrow P(j))$: wrong! (quantification over properties)
- Red has something in common with green. $\exists P(P(red) \land P(green)$: wrong! (quant. over properties of properties)

adj.
- There was a red book on the table. $\exists x(Book(x) \land Red(x) \land On\_the\_table(x))$.
- There was a small elephant in the zoo. $\exists x(Elephant(x) \land Small(x) \land In\_the\_zoo(x))$.: wrong!

adv.
- Milly swam slowly. (modifier of the verb rather than of individuals!)
- Milly swam terribly slow (modifier of a modifier).

7.1. Higher Order Logic

GQ already showed the need to move to second order (set of sets of properties).

Quantification of properties of properties brings us to third order logic etc.

Summing up, we need a **higher order logic**.

7.2. FOL: How?

Problems with the how:

Constituents: it cannot capture the meanings of constituents.

Assembly: it cannot account for meaning representation assembly.

8. 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.

We have looked at Task 1 in lecture 2 and 3 (formal grammars) and at their computational side (Implementation in Prolog, Recognition and Parsing) during the Labs.

Today we will start looking at the other two tasks.

9. Lambda Calculus

FOL augmented with Lambda calculus can capture the "how" and accomplish tasks 2 and 3.

- It has a **variable binding operators** λ. Occurrences of variables bound by λ should be thought of as place-holders for missing information: they explicitly mark where we should substitute the various bits and pieces obtained in the course of semantic construction.
- An **operation** called β-conversion performs the required substitutions.

9.1. Lambda-terms: Examples

Here is an example of lambda terms:

$$\lambda x.\texttt{left}(x)$$

The prefix $\lambda x.$ binds the occurrence of x in $\texttt{student}(x)$. We say it **abstracts** over the variable x. The purpose of abstracting over variables is to mark the slots where we want the substitutions to be made.

To glue `vincent` with "left" we need to apply the lambda-term representing "left" to the one representing "Vincent":

$$\lambda x.\texttt{left}(x)(\texttt{vincent})$$

Such expressions are called **functional applications**, the left-hand expression is called the **functor** and the right-hand expression is called the **argument**. The functor is applied to the argument. Intuitively it says: fill all the the placeholders in the functor by occurrences of the term `vincent`.

The substitution is performed by means of β-conversion, obtaining `left(vincent)`.

9.2. 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. $(\lambda x.love(x, mary))(john) \Rightarrow love(john, mary)$

9.4. Exercise

Give the lambda term representing a transitive verb.

(a) Build the meaning representation of "John saw Mary" starting from:

- John: `j`
- Mary: `m`
- saw: $\lambda x.\lambda y.\texttt{saw}(y, x)$

(b) Build the parse tree of the sentence by means the bottom-up method.

(c) Compare what you have done to assembly the meaning representation with the way you have built the tree.

9.5. α-conversion

Warning: Accidental bounds, e.g. $\lambda x.\lambda y.\mathtt{Love}(y, x)(y)$ gives $\lambda y.\mathtt{Love}(y, y)$. We need to rename variables before performing β-conversion.

α-conversion is the process used in the λ-calculus to rename bound variables. For instance, we obtain

$\lambda x.\lambda y.\mathtt{Love}(y, x)$ from $\lambda z.\lambda y.\mathtt{Love}(y, z)$.

When working with lambda calculus we always α-covert before carrying out β-conversion. In particular, we always rename all the bound variables in the functor so they are distinct from all the variables in the argument. This prevents accidental binding.

10. Practical Info

On Friday 08:30-10:30 we will continue with Lambda calculus.

On Friday 11:00-13:00 we will practice with Prolog and parsing. Hence study CFG and parsing!

Projects I have updated the web-site with papers, give a look at them an make up your mind by Friday so to start working on it next week and be ready by the 11th of November.