Logic and Natural Language Semantics: Formal Semantics

Raffaella Bernardi

DISI, University of Trento

e-mail: bernardi@disi.unitn.it

Contents

1. Logic

Question What is a Logic?

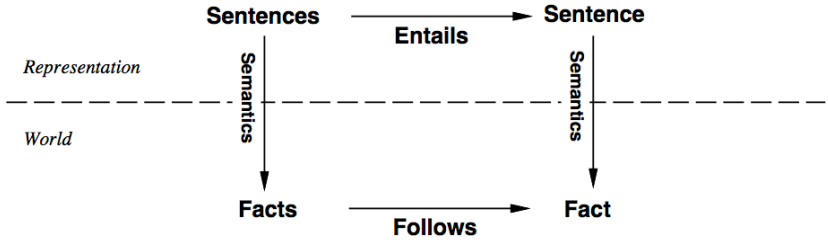
Lewis Carroll "Through the Looking Glass":

> "Contrariwise", continued Tweedledee, "if it was so, it might be; and if it were so, it would be; but as it isn't, it ain't. That's logic."

> Logic [...] is most often said to be the study of criteria for the evaluation of arguments [..], the task of the logician is: to advance an account of **valid and fallacious inference** to allow one to distinguish logical from flawed arguments.

1.1. Logic and Language

Logic in a Picture We define the language syntax and semantics and reason with it:



Question What's its connection with Language?

Aristotele's syllogisms, e.g. "All A are B", "All B are C", hence "All A are C".

Frege, Montague: a natural language can be analysed as a formal language: i.e. step by step and defining a mapping between syntax and semantics. The meaning of sentences can be represented by FOL, hence once we have sentence representations we can also reason on them with a logic systems/theorem prover.

1.2. Challenge 1: What's the meaning of linguistic signs?

Frege's question: What is identity? Is it a relation between objects or between linguistic signs?

None of the two solutions can explain why the two identities below convey different information:

(i) "Mark Twain is Mark Twain" [same obj. same ling. sign]

(ii) "Mark Twain is Samuel Clemens". [same obj. diff. ling. sign]

Frege's answer: A linguistic sign consists of a:

- **reference**: the object that the expression refers to
- **sense**: mode of presentation of the referent.

Linguistic expressions with the same reference can have different senses.

1.3. Challenge 2: From words to sentences

Complete vs. Incomplete Expressions Frege made the following distinction:

- A sentence is a **complete** expression, it's reference is the truth value.
- A proper name stands for an object and is represented by a constant. It's a **complete** expression.
- A predicate is an **incomplete** expression, it needs an object to become complete. It is represented by a function. Eg. "left" needs to be completed by "Raj" to become the complete expression "Raj left".

Principle of Compositionality: The meaning of a sentence is given by the meaning of its parts and by the compositionality rules. This holds both at the reference and sense level.

1.4. Challenge 3: Quantifiers

FOL quantifiers Frege introduced the FOL symbols: $\exists$ and $\forall$ to represent the meaning of quantifiers ("some" and "all") precisely and to avoid ambiguities.

Natural Language Syntax-Semantics The grammatical structure:

> "A natural number is bigger than all the other natural numbers."

can be represented as:

1. $\forall x \exists y Bigger(y, x)$ true
2. $\exists y \forall x Bigger(y, x)$ false

Hence, there can be a mismatch between syntactic and semantics representations

1.5. Frege: Entailment

Frege study of quantifiers, $\forall$ and $\exists$, brings to the development of FOL that can be used to represent Aristotele's syllogisms:

$\forall x(A\ x \rightarrow B\ x)$

$\forall x(B\ x \rightarrow C\ x)$

Hence, $\forall x(A\ x \rightarrow C\ x)$

But thanks to the introduction of these symbols, more complex entailment can be handled too. (We come back to this on Thursday.)

1.6. Philosophy of Language: Two lines of thoughts

Language as use Wittgenstein claims that the meaning of linguistic signs is its **use** within a context (a linguistic game made of expressions and actions), and cannot be given by a fixed set of properties since it is **vague**, but it's possible to identify the "family of expressions" to which a word/expression is similar.

Formal Semantics Important contributions to FS development are by:

1. Wittgenstein: introduces the use of Truth Tables.
2. Tarski: introduces the definition of model, domain, interpretation function and assignments that allow to treat also FOL and establish the foundation for Model Theory.
3. Montague: aims to define a model-theoretic semantics for natural language. He treats natural language as a formal language:
   - Syntax-Semantics go in parallel.
   - It's possible to define an algorithm to compose the meaning representation of the sentence out of the meaning representation of its single words.

1.7. Intermezzo: Course agenda

We will show how nowadays the two trends of philosophy of language are converging. We will introduce

- Formal Semantics as development of Frege's "reference" [Monday]
- the logic view on the natural language syntax. [Tuesday]
- the logic view on the natural language syntax-semantics interface [Wednesday]
- the non-logic view on natural language entailment [Thursday]
- Distributional Semantics as development of Frege's "sense" and Wittgenstein's "language as use" and its integration into Montague FS framework. [Friday]

2. 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 questions are closely connected.

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

2.1. 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 definitions are satisfied.

2.2. 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.

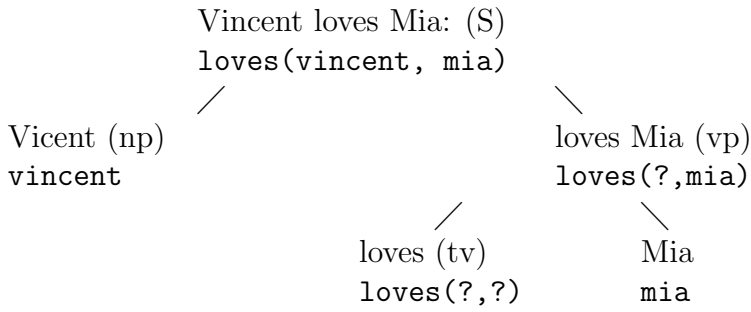
In brief, **meaning flows from the lexicon**.

Formal Semantics: How (cont'd)

But,

1 Why the meaning representation of "Vincent loves Mia" is not `love(mia,vincent)`?

The missing ingredient is the **syntactic structure**: [Vincent [loves$_v$ Mia$_{np}$]$_{vp}$]$_s$.



Briefly, **syntactic structure guiding gluing**.

2 What does "a" contribute to in "A student loves Mia"? [later]

2.3. Summing up so far: 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.

3. Meaning as Reference

Following Tarski, we build a Model by looking at a Domain (the set of entities) and at the interpretation function **interpretation function** $\mathcal{I}$ which assigns an appropriate **denotation** in the model $\mathcal{M}$ to each individual and n-place predicate constant.

Individual constants 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 $\mathcal{M}$: $\mathcal{I}(\alpha) \in \mathcal{U}$.

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 $\mathcal{U}$: $\mathcal{I}(P) \subseteq \mathcal{U}$.

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 $\mathcal{U}$: $\mathcal{I}(R) \subseteq \mathcal{U} \times \mathcal{U}$

3.1. 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.

3.2. Characteristic function

A set and **its characteristic function** amount to the same thing:

> if f_X is a function from Y to $\{F, T\}$, then $X = \{y \mid f_X(y) = T\}$. In other words, the assertion '$y \in X$' and '$f_X(y) = T$' are equivalent.

$$[\![\textit{student}]\!] = \{\text{lori, ale, sara}\}$$

`student` can be seen as a function from entities to truth values:

$$[\![\textit{student}]\!] = \{x | \texttt{student}(x) = T\}$$

Functions can be represented by lambda terms.

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

3.3. Function and lambda terms

Function $f : X \to Y$. And $f(x) = y$ e.g. $SUM(x, 2)$ if $x = 5$, $SUM(5, 2) = 7$.

The identity function $Id(x) = x$ takes a single input, x, and immediately returns x (i.e. the identity does nothing with its input).

- $\lambda x.x$
- $\lambda x.(x + 2)$ [SUM(x,2)]
- $\underbrace{(\lambda x.(x + 2))}_{function} \underbrace{5}_{argument}$
- $\underbrace{(\lambda x.(x + 2))}_{function} \underbrace{5}_{argument} = 5 + 2$

Lambda calculus was introduced by Alonzo Church in the 1930s as part of an investigation into the foundations of mathematics.

4. Lambda Calculus

- 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.
- Function can be applied to argument (**Function application**)
- An operation called **β-conversion** performs the required substitutions.
- Variables can be abstract from a term (**Abstraction**)

We will look first at how function application and β-conversion has been exploited in Formal Semantics. Then we look at abstraction too.

4.1. 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 $\texttt{left}(x)$. 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 **function** and the right-hand expression is called the **argument**. The function is applied to the argument. Intuitively it says: fill all the the placeholders in the function by occurrences of the term `vincent`.

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

4.2. Functional Application

Summing up:

- FA has the form: Function(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. $(\lambda x.love(x, mary))(john) \Rightarrow love(john, mary)$

4.3. β-conversion

Summing up:

1. Strip off the λ-prefix,
2. Remove the argument,
3. Replace all occurences of the λ-bound variable by the argument.

For instance,

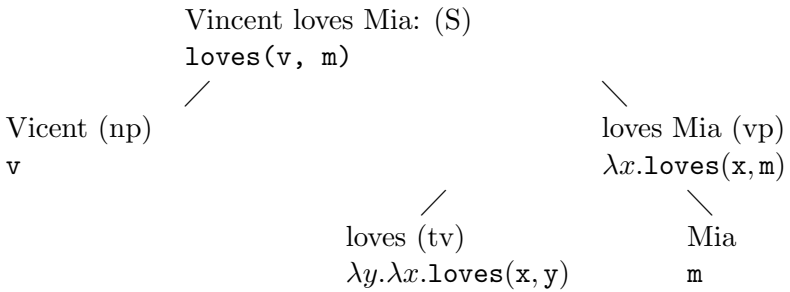
1. $(\lambda x.love(x, mary))(john)$
2. $love(x, mary)(john)$
3. $love(x, mary)$
4. $love(john, mary)$

4.4. Exercise: syntax-semantics

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

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

(b) Build the parse tree of the sentence and label it with lambda terms.



What is the set-theoretical interpretation of "loves mia"?

$$[\![\text{love mia}]\!] = \{x | love(x, m) = 1\} = \{\texttt{v}, \ldots\}$$

4.5. α-conversion

Warning: Accidental bounds, e.g. $\lambda x.\lambda y.\texttt{Love}(y, x)(y)$ gives $\lambda y.\texttt{Love}(y, y)$. 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

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

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.

4.6. λ-Terms: Models, Domains, Interpretation

In order to interpret meaning representations expressed in FOL augmented with λ, the following facts are essential:

- **Sentences**: Sentences can be thought of as referring to their truth value, hence they denote in the the domain $D_t = \{1, 0\}$.
- **Entities**: Entities can be represented as constants denoting in the domain D_e, e.g. $D_e = \{\texttt{john}, \texttt{vincent}, \texttt{mary}\}$
- **Functions**: The other natural language expressions can be seen as incomplete sentences and can be interpreted as **boolean functions** (i.e. functions yielding a truth value). They denote on functional domains $D_b^{D_a}$ and are represented by functional terms of type $(a \to b)$.

  For instance "left" misses the subject (of type e) to yield a sentence (t).

  - denotes in $D_t^{D_e}$
  - is of type $(e \to t)$,
  - is represented by the term $\lambda x_e(\texttt{left}(x))_t$

4.7. Summing up: Lambda-calculus

The pure lambda calculus is a theory of functions as rules invented around 1930 by Church. It has more recently been applied in Computer Science for instance in "Semantics of Programming Languages".

In Formal Linguistics we are mostly interested in lambda conversion and abstraction. Moreover, we work only with typed-lambda calculus and even more, only with a fragment of it.

The types are the ones we have seen above labeling the domains, namely:

- e and t are types.
- If a and b are types, then $(a \to b)$ is a type.

5. Determiners: meaning and representation

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"
  $\neg\exists x.\texttt{left}(x)$
- "Everybody left"
  $\forall x.\texttt{left}(x)$
- "Every student left"
  $\forall x.\texttt{Student}(x) \rightarrow \texttt{left}(x)$
- "A student left"
  $\exists x.\texttt{Student}(x) \wedge \texttt{left}(x)$
- "No student left"
  $\neg\exists x.\texttt{Student}(x) \wedge \texttt{left}(x)$

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

5.1. Determiners and FOL

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".

FOL does not give us the possibility to express its meaning representation. We will see now that instead lambda terms provide us with the proper expressivity.

5.2. Quantified NP and their referent

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**.

5.3. Quantified NP meaning

A QP 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"?

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

Which is the lambda terms representing the meaning of "every man"?

$$\lambda Y. \forall x. Man(x) \rightarrow Y(x)$$

5.4. Generalized Quantifiers

The study on the logical quantifier $\forall$ $\exists$ was generalized to other natural language quantifiers; hence the name "Generalized Quantifiers" (GQs) (equivalent to QPs.)

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

Try to think which are the lambda terms representing these expressions.

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

Again, which are the lambda terms representing the determiners above?

GQs have attracted the attention of many researchers working on the interaction between logic and linguistics. They have been mostly ignored by the "Language as Use" comunity.

5.5. Noun Phrases vs. Quantifier Phrases

Expressions that can be coordinated must be of the same semantic type. NPs and QPs can be coordinated, e.g.

"John and some student went to the park"

But we said that "John" and "some student" denote in D_e and $D_{(e\to t)\to t}$, resp.

Solution: We need to express that the interpretation of `john` can be taken to be the set of all the properties that hold for him:

$$[\![\texttt{john}]\!] \quad = \quad \{X \subseteq E \mid \texttt{j} \in [\![X]\!]\}.$$

To this end, linguists defined a type shifting rule: $e \Rightarrow ((e \to t) \to t)$

What is the lambda term of this new interpretation?

6. Relative Pronouns

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

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

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.

6.1. Relative Pronoun and abstraction

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: $\lambda Y.\lambda z.\texttt{read}(\texttt{j}, z) \wedge Y(z)$

Note, we use another operation of the λ-calculus: **Abstraction**!

7. Ambiguities

In natural language, there are different kinds of ambiguities:

- Lexical ambiguity, e.g. Apple
- Structural ambiguity: the same linguistic string can be structured differently
- Scope ambiguity: the same linguistic structure can receive different meaning.

7.1. Structural 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!

7.2. Scope ambiguity: QP

"Mary showed each boy an apple".

a. Then she mixed the apples up and had each boy guess which was his.

b. The apple was a MacIntosh.

This sentence has two possible meaning representations:

a. $\forall y(\texttt{Boy}(y) \rightarrow \exists x(\texttt{Apple}(x) \wedge \texttt{Show}(m, y, x)))$

b. $\exists x(\texttt{Apple}(x) \wedge \forall y((\texttt{Boy}(y) \rightarrow \texttt{Show}(m, y, x))))$

but only one syntactic structure:
[Mary [[showed [each boy]] [an apple]]]

b. is called non-local scope.

7.3. Scope ambiguity: negation

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]

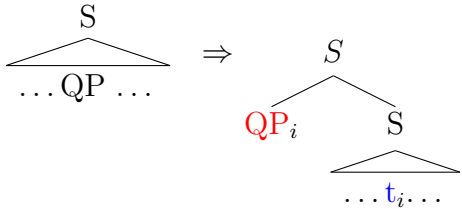
- **Scope**: In a. the quantifier phrase (QP), "a book", has scope over "didn't" [A > NOT], whereas in b. it has narrow scope [NOT > A].
- **Binding**: the variable x is bound by "a book" in "John didn't read a book".

7.4. QP: a problem for compositionality?

Expressions that can take scope in different position than where they occur

Alice [thinks [someone left]$_s$] $\exists x.think(a, left(x))$

May Quantifier Raising



A problem for compositionality? The scope operator jumps out of the syntactic structure. Putting it differently, it needs to be able "to see" the structure on which it has scope, hence assembling the meaning of structure with scope operators seems to require a "somehow" a top-down strategy.

We can back to this on Wednesday.

8. Summing up: Constituents and Assembly

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

Let's go back to the points where FOL fails, i.e. constituent representation and assembly. Problems with the how:

Constituents: it cannot capture the meanings of constituents.

Assembly: it cannot account for meaning representation 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.

9. Conclusion: Building MR

To build a meaning representation (MR) 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.

Today we have looked at Task 2 and seen that FOL augmented with Lambda calculus can capture the "how" and accomplish tasks 2 and 3.

Tuesday and Wednesday we will look at Task 1 and Task 3. We will use a non-classical logic for Task 1 and its CH Correspondence with λ-calculus for Task 3.

Thursday we'll look at successful and unsuccessful application of this approach.

Friday we will introduce an empirically based semantic model of natural language and how it can be enhanced by the syntax-semantic link studied by formal semanticists.

10. Course info

Course web site: http://disi.unitn.it/~bernardi/RSISE11/.

- Slides uploaded after each lecture.
- Bibliography
- Main conference and mailing lists

Feel free to contact me during this week or later for more info.