Logica & Linguaggio: Calcolo di Lambda III

Raffaella Bernardi

Università di Trento

P.zza Venezia, Room: 2.05, e-mail: bernardi@disi.unitn.it

Contents

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 $\lambda x.\lambda y.\texttt{read}(y, x)$.

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:

1.1. Relative Pronoun (Cont'd)

$$
\begin{array}{lcl}
[\![\texttt{lori}']\!] & = & \text{lori; } \ldots \\
[\![\lambda x.\texttt{student}'(x)]\!] & = & \{\text{lori, alex, sara}\}; \\
[\![\lambda x.\lambda y.\texttt{know}'(y,x)]\!] & = & \{(\text{lori, alex}), (\text{lori, pim}), (\text{sara, alex})\} \\
[\![\lambda \texttt{y}.\texttt{know}'(\texttt{y},\texttt{alex}')]\!] & = & \{\text{lori, sara}\} \\
[\![\lambda x.\texttt{know}'(\texttt{lori}',\texttt{x})]\!] & = & \{\text{alex, pim}\}
\end{array}
$$

The meaning of "student who lori knows" is a set of entities: {alex}.

"who" creates the intersection between the set of students and the set of those people who lori knows:

$$[\![N \ who \ VP]\!] = [\![\texttt{N}]\!] \cap [\![\texttt{VP}]\!]$$

"who": $\lambda VP.\lambda N.\lambda x.N(x) \wedge VP(x)$

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

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

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

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

2. 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".

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

Every student passed an exam?

3. 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.