1 Compositionality dilemma

Sentences with quantifiers can present an apparent mismatch between syntax and semantics. For instance, if we interpret the sentence in (1a, b) below as saying the same thing as the formula in (1c), the English quantifier “some” appears in the syntax (1b) to be in the scope of negation, but in the semantics (1c) it outscopes negation.

(1) a. Sauron didn’t find some ring.

b. 

\[
\begin{array}{c}
\text{Sauron} \\
\text{didn’t} \\
\text{find} \\
\text{some} \\
\text{ring}
\end{array}
\]

c. \[\exists x[\text{ring}(x) \& \neg \text{find}(s, x)]\]

Also, without Quantifier Substitution we cannot put together a sentence with a quantified NP in the object position. This is because of the following type mismatch:

- “some” corresponds to \(\lambda Q \eta \eta (\lambda P \eta \eta . \exists x[Q(x) \& P(x)])\)
- “ring” corresponds to \(\lambda y.\text{ring}(y)\)
- They combine via Function Application: \(\lambda Q \eta \eta (\lambda P \eta \eta . \exists x[Q(x) \& P(x)]) (\lambda y.\text{ring}(y))\)
- A \(\beta\)-conversion simplifies it to: \(\lambda P \eta \eta . \exists x[(\lambda y.\text{ring}(y))(x) \& P(x)]\)
- Another \(\beta\)-conversion gives: \(\lambda P \eta \eta . \exists x[\text{ring}(x) \& P(x)]\)
- So the QNP “some ring” corresponds to the lambda-term \(\lambda P \eta \eta . \exists x[\text{ring}(x) \& P(x)]\)
  of type \((e \to t) \to t\)
- The verb “find” corresponds to the lambda-term \(\lambda x.\lambda y[\text{find}(y, x)]\)
  of type \(e \to (e \to t)\),
- Neither the verb nor the QNP object can be an argument for the other, so they cannot combine
2 Quantifier Raising: a syntactic solution to our problem

We can solve this problem by using Quantifier Substitution, which is a purely semantic operation. Alternatively, we can use syntactic movement with traces to solve the same problem. The idea is that we compose the meaning of (1) by using the structure in (2).

(2) a. LF for (1): [some rings] \( \lambda \) Sauron didn’t find t.

\[ \text{QNP}_1 \]

\[ \text{QNP} \]

\[ \text{some rings} \]

\[ \lambda \]

\[ \text{Sauron} \]

\[ \text{didn’t} \]

\[ \text{find} \]

\[ t_i \]

Here’s how it works:

• The quantified NP raises to the top of the tree.

• This movement leaves behind a trace of type \( e \).

• **Aha!** We have resolved the type mismatch: the verb of type \( e \rightarrow (e \rightarrow t) \) can combine with the trace-object of type \( e \).

• Composition proceeds normally until the subject and VP are put together.

• The moved QNP is preceded in the tree by a lambda, waiting to catch the trace. The lambda results in a lambda-abstraction: our sentence is still missing its true object, so it is of type \( e \rightarrow t \).

• But now the last thing in the tree is the moved QNP of type \( (e \rightarrow t) \rightarrow t \). So it combines with everything normally, and the quantifier takes wide scope over the rest of the sentence.

This procedure is called Quantifier Raising, and it has been adopted by many linguists. The idea behind QR is that semantics doesn’t work directly with sentences that we hear (surface syntax), but that there is an intermediate “layer” of syntax, called the Logical Form (LF). In this class, we’ll not deal with LF, and I will not present arguments for and against its existence. You can ask me or Professor Lotus Goldberg about further readings on this debate.

References