QR recap and argument raising

Patrick D. Elliott

December 20, 2022

Contents

1 Exercises (from Coppock and Champollion)

1.1 Composing quantifiers in object position

Produce a translation into the simply-typed lambda calculus for the following sentence:

(1) Beth speaks a European language.

Here are some translations to help you get started.

- Beth $\Rightarrow$ $\mathbf{Beth} : E$
- speaks $\Rightarrow$ $\mathbf{speaks} : E \to E \to T$
- European $\Rightarrow$ $\lambda P_{ET} \,.\, \lambda x \,.\, \mathbf{european}(x) \wedge P(x) : (E \to T) \to E \to T$
- language $\Rightarrow$ $\mathbf{language} : E \to T$
- a $\Rightarrow$ $\lambda R_{ET} \,.\, \lambda S_{ET} \,.\, \exists x[P(x) \wedge Q(x)] : (E \to T) \to (E \to T) \to T$

Some important facts about quantifier raising.

- Traces are translated into *variables.*
- The moved expression introduces an *abstraction variable* into the LF, which triggers a special translation rule, *predicate abstraction.*

Predicate abstraction works as follows:

- γ is a syntax tree whose only two subtrees are x and β, where x is an abstraction variable.
- β is translated as β', an expression of type T
- Then translate γ as $\lambda x \,.\, \beta'$

Now, the exercise proper:

- Draw synax trees for the sentence (1), both before and after quantifier raising (assume that "a European language" undergoes QR). The syntax tree post-quantifier raising is called the "LF".
- Provide a translation for the sentence into the lambda calculus by compositionally translating each component part, and reducing the result using the reduction rules we've discussed.

1.2 Scope ambiguities

The following sentence is ambiguous:

(2) Some linguist offended every philosopher.

- Paraphrase the two different readings.
- Give an LF tree for each of the two readings, and specify the translation into the lambda calculus at every node of your tree.

N.b., we'll assume the following translation for *every*, the rest you should be able to figure out:

- every $\Rightarrow \lambda R_{ET} \,.\, \lambda S_{ET} \,.\, \forall x[R(x) \rightarrow S(x)] : (E \rightarrow T) \rightarrow (E \rightarrow T) \rightarrow T$

1.3 Fragment

For each of the following sentences, provide a full analysis consisting of:

- Translations for each of the lexical items.
- A syntactic tree (or more than one, if the sentence is ambiguous), where for each node you give:
  - An indication of the semantic type.
  - A full beta-reduced representation of the denotation in the lambda calculus.
  - A note of which composition rule you used.

(3) Every conservative congressman smokes.
(4) No congressman who smokes dislikes Susan.
(5) Susian respects no congressman who smokes.
(6) Susan dislikes every congressman.
(7) Some congressman from every state smokes.
(8) Every congressman respects himself.

Note you might have to get a little creative with the final two examples.

2 Quantifier scope via type-shifting

Using type-shifting rules to compose quantifiers in object position is an idea which goes back to Hendriks (1993).

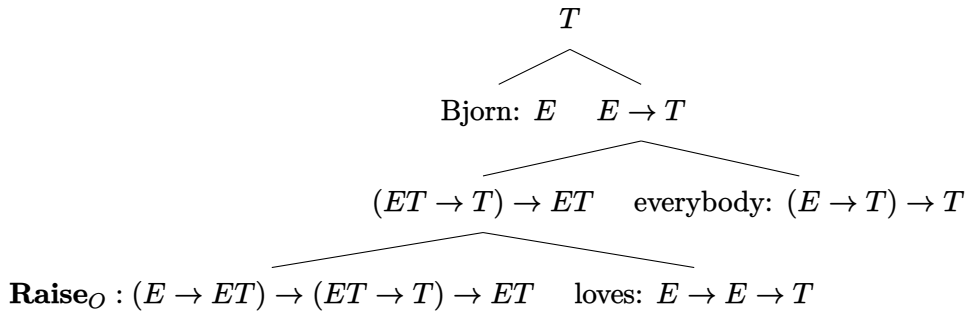
He proposes the type-shifting rule of **object raising** ($\mathbf{Raise}_O$), defined as follows:

(9) $\mathbf{Raise}_O(\alpha_{E\to\sigma T}) := \lambda Q_{ET\to T} \,.\, \lambda x_\sigma \,.\, Q(\lambda y_E \,.\, \alpha(y)(x)))$
(for any type σ)

Base on its type, how might we use object-raising to analyze the following sentence?

(10) Bjorn loves everybody.

(11)



(12) $\mathbf{Raise}_O(\mathbf{love}) = \lambda Q \,.\, \lambda x \,.\, Q(\lambda y \,.\, \mathbf{love}(y)(x))$

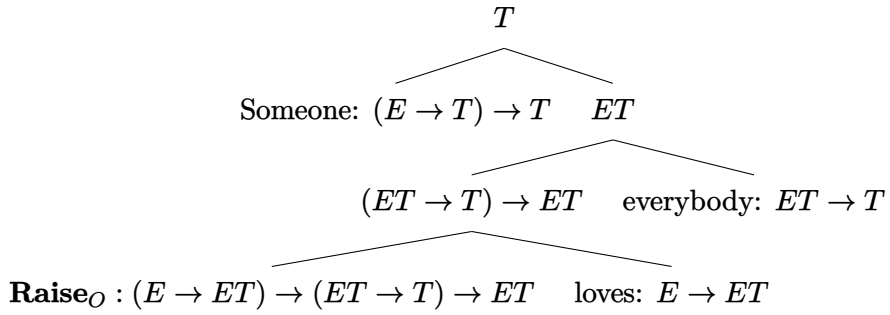
(13) $\mathbf{Raise}_O(\mathbf{love})(\mathbf{everybody}) = \lambda x \,.\, \mathbf{everybody}(\lambda y \,.\, \mathbf{love}(y)(x))$

(14) $\mathbf{Raise}_O(\mathbf{love})(\mathbf{everybody})(\mathbf{Bjorn}) = \mathbf{everybody}(\lambda y \,.\, \mathbf{love}(y)(\mathbf{Bjorn}))$

What about the following sentence? Which reading do we derive?

(15) Someone loves everybody.

(16)



(17) $\mathbf{Raise}_O(\mathbf{love}) = \lambda Q \,.\, \lambda x \,.\, Q(\lambda y \,.\, \mathbf{love}(y)(x))$

(18) $\mathbf{Raise}_O(\mathbf{love})(\mathbf{everybody}) = \lambda x \,.\, \mathbf{everybody}(\lambda y \,.\, \mathbf{love}(y)(x))$

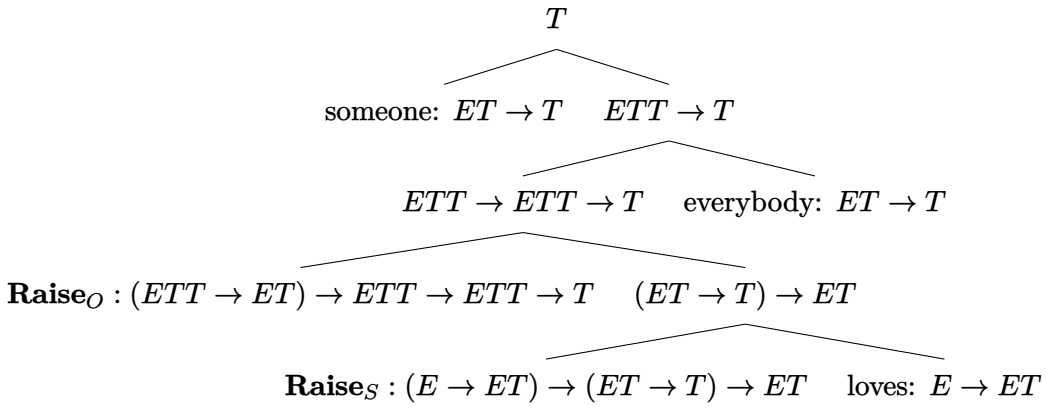
(19) $\mathbf{someone}(\mathbf{Raise}_O(\mathbf{love})(\mathbf{everybody})) = \mathbf{someone}(\lambda x \,.\, \mathbf{everybody}(\lambda y \,.\, \mathbf{love}(y)(x)))$

In order to derive the inverse scope we also need an additional type-shifting rule: subject raising.

(20) $\mathbf{Raise}_S(\alpha_{\sigma \to ET}) = \lambda y_\sigma \,.\, Q_{ET \to T} \,.\, Q(\lambda x_E \,.\, \alpha(y)(x))$

In fact, we'll need both!

(21)



(22) $\mathbf{Raise}_S(\mathbf{love}) = \lambda y \,.\, \lambda Q \,.\, Q(\lambda x \,.\, \mathbf{love}(y)(x))$

(23) $\mathbf{Raise}_O(\mathbf{Raise}_S(\mathbf{love})) = \lambda Q' \,.\, \lambda Q \,.\, Q'(\lambda y \,.\, Q(\lambda x \,.\, \mathbf{love}(y)(x)))$

(24) $\mathbf{Raise}_O(\mathbf{Raise}_S(\mathbf{love}))(\mathbf{everybody}) = \lambda Q \,.\, \mathbf{everybody}(\lambda y \,.\, Q(\lambda x \,.\, \mathbf{love}(y)(x)))$

(25) $\mathbf{Raise}_O(\mathbf{Raise}_S(\mathbf{love}))(\mathbf{everybody})(\mathbf{someone}) = \mathbf{everybody}(\lambda y \,.\, \mathbf{someone}(\lambda x \,.\, \mathbf{love}(y)(x)))$