Quantifier raising

Handout 3

Patrick D. Elliott

November 22, 2022

Contents

1 Recap

Reading assigned last time:

- *Invitation to Formal Semantics*
  - chapter 6.3-6.4
  - Chapter 7.4.1

1.1 Quantificational NPs

Quantifificational NPs are of type $(E \rightarrow T) \rightarrow T$.

(1) $\textbf{everyone} : (E \rightarrow T) \rightarrow T$

Quantifiers can compose with VPs via *functional application.*

(2)
$$\begin{tikzcd}[column sep=tiny] & \textbf{everyone}(\textbf{left}) : T \arrow[dl, dash] \arrow[dr, dash] & \\ \textbf{everyone} : (E \rightarrow T) \rightarrow T & & \textbf{left} : E \rightarrow T \end{tikzcd}$$

They express *higher-order functions* from functions to truth values.

We can translate freely from functions of type $\sigma \rightarrow T$ to sets of σs using the following operation:

(3) $Set(f) = \{\, x \mid f(x) = \textbf{true} \,\}$

From a set-theoretic perspective, quantifiers express *functions from sets of individuals to truth-values* - or, to go one step further *sets of sets of individuals.*

1.1.1 First order quantifiers

We can elucidate what "first order" quantificational NPs do using propositional logic:

(4) $\textbf{everybody} := \lambda P \,.\, \forall x[P(x)]$

(5) $\textbf{somebody} := \lambda P \,.\, \exists x[P(x)]$

(6) $\textbf{nobody} := \lambda P \,.\, \neg\exists x[P(x)]$

- **Question:** give a denotation for each of the above quantifiers as a set of sets of individuals.

In order to deal with quantificational NPs which are clearly decompositional, we need to give a translation for quantificational *determiners* such as *every*, *some* and *no*:

(7) $\textbf{every} := \lambda R \,.\, \lambda P \,.\, \forall x[R(x) \rightarrow P(x)]$

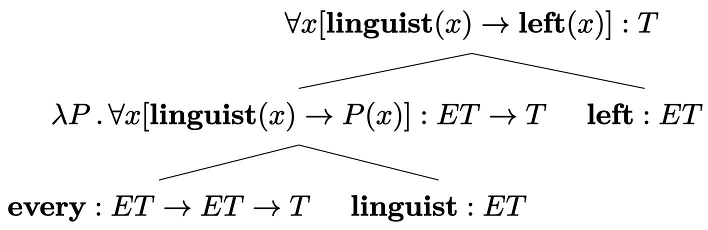
(8) $\textbf{some} := \lambda R \,.\, \lambda P \,.\, \exists x[R(x) \wedge P(x)]$

(9) $\textbf{no} := \lambda R \,.\, \lambda P \,.\, \neg\exists x[R(x) \wedge P(x)]$

Determiners are expressions of type $(E \to T) \to (E \to T) \to T$.

We compose progressively via function application just as before:

(10)


$\forall x[\mathbf{linguist}(x) \to \mathbf{left}(x)] : T$
$\lambda P\,.\,\forall x[\mathbf{linguist}(x) \to P(x)] : ET \to T$
$\mathbf{left} : ET$
$\mathbf{every} : ET \to ET \to T$
$\mathbf{linguist} : ET$

1.2 Quantifiers scoping over disjunction

Exercise 16, from the Coppock and Champollion textbook.

(11) Everybody smokes or drinks.

- Is a ‘conjunction reduction’ analysis plausible?
- Assume the following entry for “or” and show how this solves the problem. It might be useful to think about the meaning of *smokes or drinks* and *everybody* in set terms.

(12) $\mathbf{or} := \lambda P\,.\,Q\,.\,\lambda x\,.\,Q(x) \vee P(x)$ $\qquad ET \to ET \to ET$

(Partee & Rooth 1983) show that the translation of **or** given above is a recursive generalization of boolean disjunction.

(We’ll actually come back to what this actually means when we cover continuation semantics).

2 Generalized quantifiers

Generalized quantifiers are the sets of sets encoded by the functions denoted by quantificational NPs:

(13) $\mathbf{every}(\mathbf{cat}) = \lambda P\,.\,\forall x[\mathbf{cat}(x) \to P(x)]$

(14) $\{\, P \subseteq \mathbf{Dom}_E \mid \{\, x \mid \mathbf{cat}(x) \,\} \subseteq P \,\}$

(15) $\mathbf{some}(\mathbf{dog}) = \lambda P\,.\,\exists x[\mathbf{dog}(x) \wedge P(x)]$

(16) $\{\, P \subseteq \mathbf{Dom}_E \mid P \cap \{\, x \mid \mathbf{dog}(x) \,\} \neq \emptyset \,\}$

(17) $\mathbf{everything} = \lambda P\,.\,\forall x[P(x)]$

(18) $\{ P \subseteq \mathbf{Dom}_E \mid \mathbf{Dom}_E \subseteq P \} = \{ \mathbf{Dom}_E \}$

(19) $\mathbf{something} = \lambda P \,.\, \exists x[P(x)]$

(20) $\{ P \subseteq \mathbf{Dom}_E \mid \mathbf{Dom}_E \cap P \neq \emptyset \} = \{ P \subseteq \mathbf{Dom}_E \mid P \neq \emptyset \}$

(21) $\mathbf{nothing} = \lambda P \,.\, \neg\exists x[P(x)]$

(22) $\{ P \subseteq \mathbf{Dom}_E \mid P = \emptyset \} = \{ \emptyset \}$

Generalized quantifiers are more expressive than first order logic.

(23) $\lambda P \,.\, \mathbf{exactlyTwoThings}(P)$

(24) $\{ P \subseteq \mathbf{Dom}_E \mid \mathbf{Card}(P) = 2 \}$

(25) $\lambda P \,.\, \mathbf{atLeastTwoThings}(P)$

(26) $\{ P \subseteq \mathbf{Dom}_E \mid \mathbf{Card}(P) \geq 2 \}$

Famously, proportional quantifiers such as *most things* can't be defined in first order logic at all:

(27) $\lambda P \,.\, \mathbf{most}(\mathbf{swans})(P)$

(28) $\{ P \subseteq \mathbf{Dom}_E \mid \mathbf{Card}(\{ x \mid \mathbf{swan}(x) \} \cap P) \geq \mathbf{Card}(\{ x \mid \mathbf{swan}(x) \} - P) \}$

In set terms, determiners express *curried relations between sets*:

(29) $\mathbf{every} := \lambda R \,.\, \lambda P \,.\, \forall x[R(x) \rightarrow P(x)]$

(30) $\{ (R, P) \mid R \subseteq P \}$

(31) $\mathbf{some} := \lambda R \,.\, \lambda P \,.\, \exists x[R(x) \wedge P(x)]$

(32) $\{ (R, P) \mid R \cap P \neq \emptyset \}$

- The R argument of a determiner is called its **restrictor**.
- The P argument of a determiner is called its **nuclear scope**.

There are some logical properties that all natural language determiners seem to have in common, such as **conservativity**: *every A that Bs* is always equivalent to *every A is an A that Bs* (for *every* as well as any other putative quantifier). see the Coppock and Champollion textbook for more discussion on this point.

3 Modification

3.1 Adjectival modification

(33) Louise is a happy dog.

(34) $\Rightarrow$ Louise is happy.

(35) $\Rightarrow$ Louise is a dog.

(36) $\mathbf{happy} : E \to T$

(37) $\mathbf{dog} : E \to T$

A freely available type-shifting operator $\wedge : ET \to ET \to ET$

(38) $\wedge(P) := \lambda Q \,.\, \lambda x \,.\, P(x) \wedge Q(x)$

(39)

$$\begin{tikzcd}
 & \mathbf{happy}(\mathbf{Louise}) \wedge \mathbf{dog}(\mathbf{Louise}) : T \arrow[dl, no head] \arrow[dr, no head] & & \\
\mathbf{Louise} : E & & \lambda x \,.\, \mathbf{happy}(x) \wedge \mathbf{dog}(x) : ET \arrow[dl, no head] \arrow[dr, no head] & \\
 & \lambda P \,.\, \lambda x \,.\, \mathbf{happy}(x) \wedge P(x) : ET - ET \arrow[d, no head] & & \mathbf{dog} : E \to T \\
 & \wedge(\mathbf{happy}) : ET \to ET & &
\end{tikzcd}$$

3.2 Relative clauses

(40) dog that Josie loves.

This expression characterizes an individual with two properties:

- She is a dog.
- Josie loves her.

In other words, this is (the characteristic function of) the intersection between the set of individuals that are dogs, and the set of individuals loved by Josie.

We need to somehow translate *that Josie loves* as follows:

(41) $\lambda x \,.\, \mathbf{loves}(x)(\mathbf{Josie}) : E \rightarrow T$

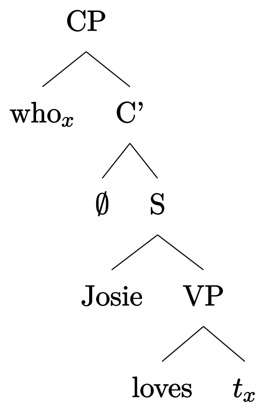
The strategy we'll pursue is to draw an analogy between the following:

(42) dog that Josie loves.

(43) dog such that Josie loves her.

The *that* or *who* of the relative clause *moves*, and leaves behind a silent pronoun (called a trace) in its "base" position.

(44)



We assume that the syntax pairs the moved expression *who* and its trace with matching variables - this will be important for the purposes of interpretation.

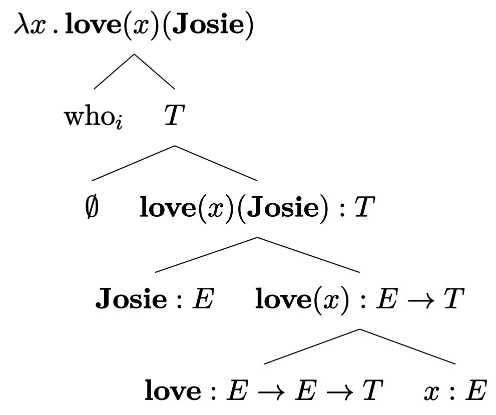
We'll assume that traces are translated directly as variables. So, for example *Josie loves* t_x is translated as:

(45) $\mathbf{loves}(x)(\mathbf{Josie})$

In order to ensure that the variable gets bound by a matching lambda operator we (sadly) need a special, syncategorematic rule for translating structures involving binding operators. This special translation rule is called **Predicate Abstraction**.

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

(46) Predicate abstraction


$\lambda x \,.\, \mathbf{love}(x)(\mathbf{Josie})$
who$_i$
T
$\emptyset$
$\mathbf{love}(x)(\mathbf{Josie}) : T$
$\mathbf{Josie} : E$
$\mathbf{love}(x) : E \to T$
$\mathbf{love} : E \to E \to T$
$x : E$

In order to account for cases without an overt *wh*-operator, i.e., relative clauses formed with *that*, or with nothing at all such as “the dog Josie loves”, it’s common to post a silent counterpart of *who*$_x$: Op_x.

(47) boy [Op_x Josie loves t_x]

We can account for the composition of the relative clause and noun by invoking the same conjunctive type-shifter which we used for adjectival modification.

(48) $\wedge(\lambda x \,.\, \mathbf{love}(x)(\mathbf{Josie}))(\mathbf{dog})$

- **Exercise:** verify that this gives back the right results.

3.3 Locality

You might be skeptical about this analysis, especially when Op_i is implicated, but relative clause formation shows many of the same restrictions, as e.g., question formation in English.

(49) *The boy [Op_x every friend of t_x insulted].

(50) *Who$_x$ did every friend of t_x insult?

(51) *The boy [Op_x Sarah insulted t_x and praised Josie].

(52) *Who$_x$ did Sarah insult t_x and praise Josie?

These constraints on movement are known as **island constraints** (Ross 1967).

The fact that relativization and *wh*-movement are subject to the same constraints suggests that they should receive a unified analysis (Chomsky 1977).

One interesting fact about this kind of movement is that it can proceed an arbitrary distance out of embedded environments.

(53) the boy [Op_x Sarah believes [Josie insulted t_x]].

(54) Who$_i$ does Sarah believe [Josie insulted t_x]?

(55) the boy [Op
x Sarah told me [that Sam believes [Josie insulted t_x]]]

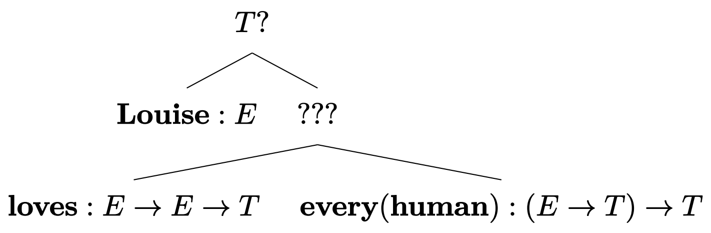
(56) Who$_x$ did Sarah told me [that Sam believes [Josie insulted t_x]]?

- We don't have a good treatment of the semantics of verbs like *believes*, but for the time being let's assume that it's of type $T \to E \to T$.
- Show how composition proceeds in the first example, above.

Once we introduce *movement* in order to understand certain sentences with quantifiers, we'll return to this concept. It will be one of our initial motivations for moving over to continuation semantics. But first...

4 Quantifiers in object position

A problem! Quantificational NPs can't compose when they're in any position other than subject position.

(57)

T?
Louise : E
???
loves : $E \to E \to T$
every(**human**) : $(E \to T) \to T$

Remember that *every human* has the following meaning (roughly):

(58) $\lambda P\,.\,\forall x[\mathbf{human}(x) \to P(x)]$

In this case, an appropriate value for P would be one which maps an individual to true if Louise loves them, i.e.:

(59) $\lambda x \,.\, \mathbf{love}(x)(\mathbf{Louise})$

We derive this component via a new syntactic transformation: *Quantifier raising.*

Recall the special translation rule **Predicate abstraction**.

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

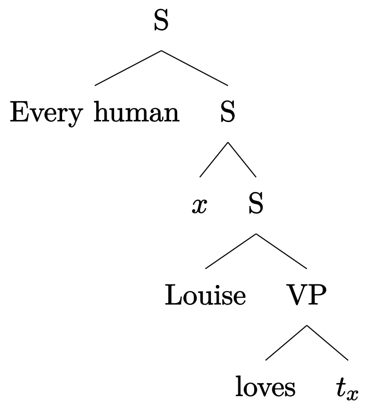
We'll assume that any NP can be supplied with a variable by the syntax, and may move.

We'll need to modify the translation rule slightly however, since NPs do more than just induce abstraction:

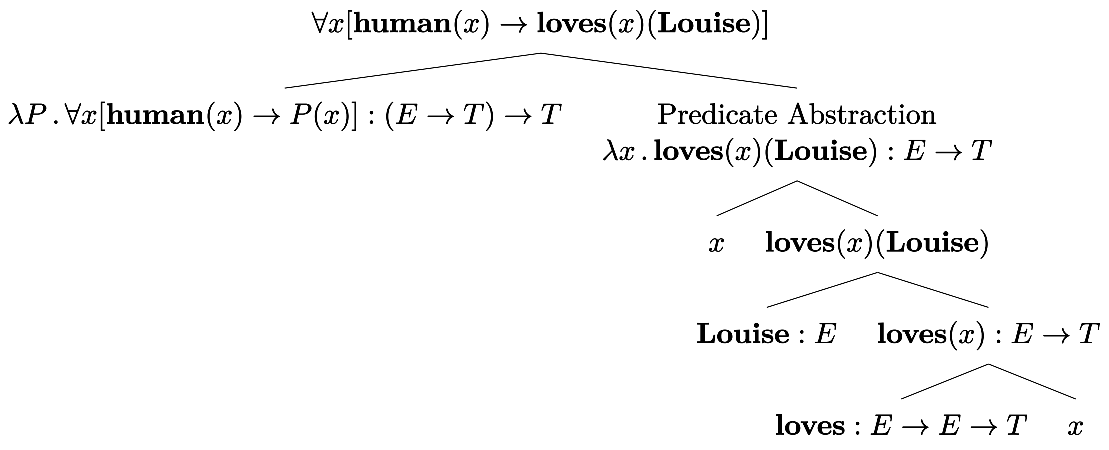
- γ is a syntax tree whole only two subtrees are x and β, where x is a variable.
- β is translated as β', an expression of type T
- Then translate γ as $\lambda x \,.\, \beta'$

Following (Heim & Kratzer 1998), we assume that movement of an NP involves rebracketing of the associated variable with the sister of the moved NP.

(60)

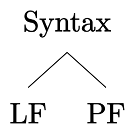


(61)


$\forall x[\mathbf{human}(x) \to \mathbf{loves}(x)(\mathbf{Louise})]$
$\lambda P\,.\,\forall x[\mathbf{human}(x) \to P(x)] : (E \to T) \to T$
Predicate Abstraction
$\lambda x\,.\,\mathbf{loves}(x)(\mathbf{Louise}) : E \to T$
x
$\mathbf{loves}(x)(\mathbf{Louise})$
$\mathbf{Louise} : E$
$\mathbf{loves}(x) : E \to T$
$\mathbf{loves} : E \to E \to T$
x

The idea of Quantifier raising was originally formulated in a syntactic theory with several levels of representation (May 1977):

- Narrow syntax (sometimes split into deep and surface structure)
- Logical Form (the input to semantic interpretation)
- Phonological Form (the input to pronunciation)



Quantifier raising is an operation that takes place on the branch from Syntax to LF, therefore it has semantic consequences but not phonological consequences.

Other operations, like *wh*-movement, and relative clause formation, take place in the syntax, and therefore have both semantic *and* phonological consequences.

4.1 Scope ambiguities with operators

(62) John hasn't read exactly three books.

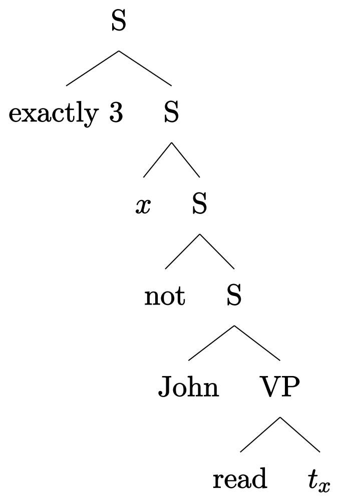
1. *The cardinality of books that John hasn't read is 3.* Can be true, for example, if John has read 3 of the books, and hasn't read 3 of the books.
2. *It's not the case that the cardinality of books that John has read is 3.* Can be true, for example, if John has read 4 of the books and hasn't read 2 of them.

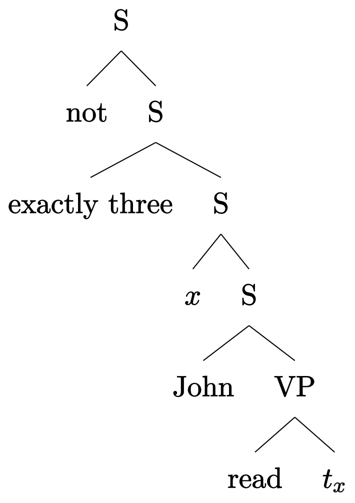
Note that, in scenario 1, the first reading is true, and the seconds is false; in scenario 2; the first reading is false, and the second reading is true, so these readings are semantically independent.

(63) $\mathbf{Card}(\{\,x \mid \mathbf{book}(x)\,\} \cap \{\,x \mid \neg\mathbf{read}(x)(\mathbf{John})\,\}) = 3$

(64) $\mathbf{Card}(\{\,x \mid \mathbf{book}(x)\,\} \cap \{\,x \mid \mathbf{read}(x)(\mathbf{John})\,\}) \neq 3$

The possibility of quantifier raising can be used to derive these two readings.


S
exactly 3
S
x
S
not
S
John
VP
read
t_x


S
not
S
exactly three
S
x
S
John
VP
read
t_x