Generalized Quantifiers & Categorial Approaches & Intensionality

Last week

- The semantics of questions is hard to characterize directly. Instead, we usually start with the relation between questions, answers, and statements. We also consider other “question-containing” constructions (e.g. free relatives, wh-conditionals).

- The principle of compositionality  The meaning of a complex expression is determined by the meanings of its parts and the way they are syntactically combined.

- Types

  (1)  a. Basic types: e (individuals/entities) and t (truth values).
       b. Functional types: If σ and τ are types, then ⟨σ, τ⟩ is a type.

- Syntactic categories and their semantic types

<table>
<thead>
<tr>
<th>Syntactic category</th>
<th>English expressions</th>
<th>Semantic type (extensionalized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence</td>
<td></td>
<td>t</td>
</tr>
<tr>
<td>Proper name</td>
<td>John</td>
<td>e</td>
</tr>
<tr>
<td>e-type/referential NP</td>
<td>the king</td>
<td>e</td>
</tr>
<tr>
<td>Common noun</td>
<td>cat</td>
<td>⟨e, t⟩</td>
</tr>
<tr>
<td>IV, VP</td>
<td>run</td>
<td>⟨e, t⟩</td>
</tr>
<tr>
<td>TV</td>
<td>love</td>
<td>⟨e, et⟩</td>
</tr>
<tr>
<td>Predicative ADJ</td>
<td>happy, gray</td>
<td>⟨e, t⟩</td>
</tr>
<tr>
<td>Predicate modifier</td>
<td>skillful, quickly</td>
<td>⟨et, et⟩</td>
</tr>
<tr>
<td>Sentential modifier</td>
<td>perhaps</td>
<td>⟨t, t⟩</td>
</tr>
<tr>
<td>Determiner</td>
<td>some, every, no</td>
<td>⟨et, ⟨et, t⟩⟩</td>
</tr>
<tr>
<td></td>
<td>the, a</td>
<td>⟨et, e⟩</td>
</tr>
</tbody>
</table>

- With type assignments to expressions of natural language, we can determine the semantic types of new expressions/morphemes. But, we need to be careful: an expression can be type-ambiguous; there can be covert elements in the LF; there can be type-shifting operations, ....

- It is handy and common to write functions in lambda (λ)-notations.

  (2) Schema:  \( \lambda v [\beta, \alpha] \)  read as “the function which maps every \( v \) such that \( \beta \) to \( \alpha \)”

  (3) Semantic types:

    If \( v \) is of type \( \sigma \) and \( \alpha \) is of type \( \tau \), then \( \lambda v. \alpha \) is of type \( \langle \sigma, \tau \rangle \).

  (4) Lambda reduction

    \( (\lambda v. \alpha)(a) = \alpha' \) where \( \alpha' \) is like \( \alpha \) but with every free occurrence of \( v \) replaced by \( a \).
• Composition rules

(5) Terminal Nodes (TN)
If $\alpha$ is a terminal node, $[\alpha]$ is specified in the lexicon.

Non-Branching Nodes (NN)
If $\alpha$ is non-branching node, and $\beta$ is its daughter node, then $[\alpha] = [\beta]$.

Functional Application (FA)
If $\{\beta, \gamma\}$ is the set of $\alpha$’s daughters, $[\beta] \in D_{\langle \sigma, x \rangle}$, and $[\gamma] \in D_{\sigma}$, then $[\alpha] = [\beta](\gamma)$

Predicate Modification (PM)
If $\alpha$ is a branching node, $\{\beta, \gamma\}$ is the set of $\alpha$’s daughters, and $[\beta]$ and $[\gamma]$ are both in $D_{\langle \sigma, x \rangle}$, then $[\alpha] = \lambda x_{\sigma}[\beta](x) = [\gamma](x) = 1$

• Warning: Take a meaning as a whole!
For any two meanings $\alpha$ and $\beta$, the combination of $\alpha$ and $\beta$ can only depend on what $\alpha$ and $\beta$ are, each “taken as a whole”; it cannot depend on the meanings that $\alpha$ and $\beta$ were formed from by earlier semantic operations.

Plan for today
• Generalized quantifiers
• Categorial approaches of question semantics
• Intensionality
1 Generalized quantifiers

This part is crucial for studying Karttunen Semantics, \textit{wh}-movement, and quantifying-into questions.

1.1 Generalized quantifiers

- Quantificational DPs (e.g. \textit{everything}, \textit{something}, \textit{every cat}, \textit{some cat}) are not individuals (cf. proper names like \textit{John}), nor individual sets (cf. common nouns like \textit{cat}).

E.g. Only an \(e\)-type NP can normally license a singular discourse pronoun.

(6) a. John /the man/ a man walked in. He looked tired.
   b. Every man /no man/ more than one man walked in. *He looked tired.

We treat quantificational DPs as second-order functions of type \(\langle et, t \rangle\), called “generalized quantifiers (GQs)”. In (7), \textit{meows} is an argument of \textit{every cat}.

(7) \[
\begin{array}{c}
S_t \\
| \\
\text{DP}_{\langle et, t \rangle} | \\
| \\
\text{VP}_{\langle et, t \rangle} \\
| \\
\text{D} | \\
\text{NP} \\
| \\
\text{every cat}_{\langle et, t \rangle} \\
\end{array}
\]

(8) a. \([\text{every cat}] = \lambda P_{\langle et, t \rangle}. \forall x [\text{cat}'(x) \rightarrow P(x)]\]
   b. \([\text{some cat}] = \lambda P_{\langle et, t \rangle}. \exists x [\text{cat}'(x) \land P(x)]\]

- The determiner \textit{every} combines with a common noun of type \(\langle e, t \rangle\) to return a generalized quantifier of type \(\langle et, t \rangle\). Therefore, its type is quite complex: \(\langle et, \langle et, t \rangle \rangle\).

(9) a. \([\text{every}] = \lambda Q_{\langle et, t \rangle}. \lambda P_{\langle et, t \rangle}. \forall x [Q(x) \rightarrow P(x)]\]
   b. \([\text{some}] = \lambda Q_{\langle et, t \rangle}. \lambda P_{\langle et, t \rangle}. \exists x [Q(x) \land P(x)]\]

- In comparison, a \textit{the}-phrase denotes a referential individual of type \(e\). The determiner \textit{the} is of type \(\langle et, t \rangle\): it combines with a common noun of type \(\langle e, t \rangle\) to return a referential individual of type \(e\).

(10) a. \([\text{the cat}] = \iota x [\text{cat}'(x)]\) (To be revised)
   b. \([\text{the}] = \lambda P. \iota x [P(x)]\) (To be revised)

- Individuals (of type \(e\)) can also be shifted into generalized quantifiers via \textit{type-lifting}.

(11) a. \([\text{John}] = j\)
   b. \text{LIFT}([\text{John}]) = \lambda P_{\langle et, t \rangle}. P(j)
   c. \text{LIFT}([\text{John}]) ([\text{came}]) = (\lambda P_{\langle et, t \rangle}. P(j))(\lambda x. \text{came}'(x))
      = (\lambda x. \text{came}'(x))(j) = \text{came}'(j)
1.2 Be-shifting

- We can extract the quantification domain of an $\exists$-quantifier via the BE-shifter (Partee 1986):

\begin{align*}
(12) \quad \text{BE} &= \lambda \mathcal{P} \lambda z [P(\lambda y.y = z)] \\
(13) \quad \text{BE}([\text{some cat}]) &= \lambda z [(\lambda f_{e,t}, \exists x \in \text{cat'}[f(x)])(\lambda y.y = z)] \\
&= \lambda z [\exists x \in \text{cat'}[x = z]] \\
&= \{z : z \in \text{cat'}\} \\
&= \text{cat'}
\end{align*}

- Discussion: What do we get by applying BE to [every cat] and LIFT([John])?

1.3 Quantifier raising

- Problem in (14a): a type-mismatch arises when a verb takes a quantificational DP as its object.

Solution in (14b): At logical form (LF), the generalized quantifier can raise to adjoin to any propositional node, leaving a trace of type $e$ and introducing a lambda abstraction over the traces variable. This operation is called “Quantifier Raising (QR)”.

(14) Anna loves every cat.

a. $\text{S}$
   
   Anna
   
   $\lambda x \text{loves}_{e,t}$
   
   $\langle e,t \rangle$

b. $\text{S}$
   
   $\langle e,t \rangle$
   
   every cat
   
   $\lambda x$
   
   Anna
   
   $\lambda x \text{loves}_{e,t}$
   
   $\langle e,t \rangle$

- Scope ambiguity can be characterized based on QR.

(15) Every kid loves some cat.

a. Surface scope reading: $\forall x [\text{kid}'(x) \rightarrow \exists y [\text{cat'}(y) \wedge \text{love'}(x, y)]]$

b. Inverse scope reading: $\exists y [\text{cat'}(y) \wedge \forall x [\text{kid}'(x) \rightarrow \text{love'}(x, y)]]$

1. Exercise: Draw a tree to represent the LF of the following sentence.

(16) We require no student to come to the office hour.
• In generative grammar, lambda-abstraction is to represent phrasal movement. The sister node of an λ-abstract is the moved phrase. The variable bound by the λ-operator is the trace.

Example: using lambda-abstraction to represent subject movement:

(17) Mary will come.

• Discussion: The following LFs are problematic. Identify the problems.

(18) Anna loves every cat.

(19) Every kid loves some cat.

• Discussion: Hamblin-Karttunen semantics defines a question as a set of propositions. Under this view, is the following LF well-formed? Why or why not?

(20) Which cat does every kid love?  
    ≈ For every kid x, which cat does x love?
2 Categorial approaches of question semantics

- A wh-question can receive short or full answers.

(21) Who came?
   a. John. (short answer)
   b. John came. (full answer)

- Core assumptions in categorial approaches\(^1\)
  - short answers are bare nominal, not covertly clausal (cf. Merchant 2004).
  - short answer is primary; the root denotation of a question is a function (or a lambda abstract) that can take a denotation of a short answer as an argument and return the denotation of the corresponding full answer.

(22) a. \([\text{who came}] = \lambda x[\text{people}')(x).\text{came}'(x)]\]
    b. \(\lambda x[\text{people}')(x).\text{came}'(x)](j) = \text{came}'(j)\)

- Wh-items are lambda operators.

(23) a. \([\text{who}] = \lambda P\lambda x[\text{people}')(x).P(x)]\]
    b. \([\text{what}] = \lambda P\lambda x[\text{thing}')(x).P(x)]\]

Examples:

(24) Who came?
    (25) What did John buy?

\[\text{Who} \lambda x \text{IP} \text{x came}\]
\[\text{What} \lambda x \text{IP} \text{John bought x}\]

- Discussion: Do you remember the difficulty with deriving the following denotation compositionally?

(26) \([\text{who bought what}] = \lambda x\lambda y[\text{people}')(x) \land \text{thing}'(y).\text{bought}'(x, y)]\]

- Advantages of categorial approaches

Ad1: The relation between questions and short answers is very directly modeled:

Question (short answer) = Function (argument)

Ad2: Similarity of wh-questions and free relatives is captured nicely.

(27) a. “Whom did Mary vote for?” “Andy and Billy.”
    b. We hired whom Mary voted for. = We hired Andy and Billy.

(28) a. “Where can we get coffee?” “Starbucks”/“J.P. Licks”/...
    b. We went to where can get coffee. = We went to Starbucks/J.P. Licks/...

• Why it is necessary to model the relation between questions and short answers semantically? (wait till week 11)

• Why is advantageous to unify questions and free relatives?

Caponigro (2003, 2004) observes that *wh*-words are cross-linguistically more restrictively distributed in free relatives than in questions.

(29) **Caponigro’s Generalization** (Caponigro 2003, 2004)

If a language uses the *wh*-strategy to form both questions and free relatives, the *wh*-words found in free relatives are always a subset of those found in questions. Never the other way around. Never some other arbitrary relation between the two sets of *wh*-words.

⇒ the derivation of a question is strictly simpler than that of the corresponding free relative. (Otherwise, there would be some *wh*-constructions that could be used as free relatives but not as questions, contra the generalization.)

⇒ Most likely, free relatives are derived from questions (Chierchia & Caponigro 2013). C’s generalization is predicted as long as the operation from questions to free relatives is partial.

![Diagram](free relatives Partial Op Questions)

• Problems of categorial approaches

**P1:** It assigns different semantic types to different questions, which makes it difficult account for question coordinations:

(30) a. John asked Mary [[(_e,t) who came] and [(_e,et) who bought what]].
    b. John knows [[(_e,t) who came] and [(_e,et) who bought what]].

**P2:** Treating *wh*-items as lambda operators cannot account for the cross-linguistic fact that *wh*-words behave like existential indefinites in non-interrogatives.

(31) Mandarin: *shenme*-NP

a. Yuehan haoxiang jian-le *shenme-ren*
   John perhaps meet-PERF what-person
   ‘It seems that John met **someone**.’

b. Ruguo Yuehan jian-guo *shenme-ren,* qing gaosu wo.
   If John meet-EXP what-person, please tell me.
   ‘If John met **someone**, please tell me.’
• Solving the problems (Xiang 2016: ch. 1)

For P1: A question coordination is a generalized quantifier. It undertakes QR and moves to left edge of the matrix clause, yielding a wide scope reading of and relative to the matrix verb know:

John knows who came and John knows who bought what.

\[ \text{John knows } [Q_1 \text{ who came}] \text{ and } [Q_2 \text{ who bought what}]. \]

For P2: A wh-item denotes an existential quantifier. In a question, it is converted into a domain restrictor via a BE\text{DOM}-shifter.

\[ \text{(33) Definition: BE\text{DOM}} \]
\[ \text{BE\text{DOM}}(P) = \lambda \theta, t P, [\theta : P, \text{BE}(P)] \land \forall \alpha \in \text{Dom}(P)[P(\alpha) = \theta(\alpha)] \]

(\text{BE\text{DOM}}(P) \text{ applies to a function } \theta \text{ (of an arbitrary type } \tau) \text{ and restricts the domain of } \theta \text{ with the quantification domain of } P. \)

Example (for now, let’s ignore intensionality):

\[ \text{(34) Which boy came?} \]

Using this approach, we can also easily derive the denotation of multi-wh questions.
3 Intensionality

3.1 Extensional Semantics is not enough

- In Extensional Semantics, every expression is of type \( e \), or \( t \), or a derived type based on \( e \) and \( t \).
  - Sentence:
  - Common NP:
  - Intransitive verb:

But, Extensional Semantics is insufficient for modeling question semantics. For example, Hamblin semantics treats a question as denoting a set of possible answers; if using only Extensional Semantics, Hamblin Semantics predicts the following:

\[
\begin{align*}
(35) \quad & \text{a. } [\text{Did Mary come?}] = \{[\text{Mary came}], [\text{Mary didn’t come}]\} = \{1, 0\} \\
& \text{b. } [\text{Did John come?}] = \{[\text{John came}], [\text{John didn’t come}]\} = \{1, 0\} \\
& \rightarrow \text{Yes-no questions all have the same semantics, namely, } \{1, 0\}. \text{ NO WAY!}
\end{align*}
\]

3.2 Defining extension and intension

- The extension of an expression is dependent on the evaluation world. We add an evaluation world parameter \([\bullet]^w\) to the notations of extensions:

\[
(36) \quad \text{General notation: } [X]^w \quad (\text{the extension of } X \text{ in } w')
\]

Examples:

\[
(37) \quad \begin{align*}
\text{a. } & [\text{Mary came}]^w = 1 \text{ iff Mary came in } w. \\
\text{b. } & [\text{old}]^w = \lambda x. x \text{ is old in } w. \\
\text{c. } & [\text{bachelor}]^w = \\
\text{d. } & [\text{hit}]^w =
\end{align*}
\]

- The intension of an expression \( X \) is a function which (i) takes a possible world as an argument, and (ii) returns the extension of \( X \) in that world.

\[
(38) \quad \text{General notation: } \lambda w. [X]^w \quad (\text{the intension of } X')
\]

- The intension of a sentence is a function from worlds to truth values, called proposition.
- The intension of a predicate (IV/VP/NP/Pred Adj/..) of type \( \langle e, t \rangle \) is a function from worlds to \( \langle e, t \rangle \) functions, called property.
- The intension of a definite NP is a function from worlds to entities, called individual concept.

Examples: (the descriptions of each example are all equivalent)

\[
\begin{align*}
(39) \quad & \text{The intension of “Mary came”:} \\
& \quad \begin{align*}
\text{a. } & \lambda w. \text{Mary came in } w \\
\text{b. } & \lambda w. [\text{Mary came}]^w = 1 \\
\text{c. } & \lambda w. [\text{Mary came}]^w \\
\text{d. } & ...
\end{align*} \\
(40) \quad & \text{The intension of “the king”:} \\
& \quad \begin{align*}
\text{a. } & \lambda w. \text{the unique king in } w \\
\text{b. } & \lambda w. \lambda x. [\text{king}^w(x) = 1] \\
\text{c. } & \lambda w. \lambda x. [\text{king}^w(x)] \\
\text{d. } & ...
\end{align*}
\end{align*}
\]
• Propositions can also be viewed as the set of possible worlds where this proposition is true.

(41) The intension of “Mary came”:
   a. \{w : \text{Mary came in } w\}
   b. \{w : [\text{Mary came}]^w = 1\}

Hence, we can use set-theoretical operations to represent the following relations and operations:

<table>
<thead>
<tr>
<th>Relations and operations</th>
<th>Set-theoretical notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p) entails (q)</td>
<td>(p \subseteq q)</td>
</tr>
<tr>
<td>(p) contradicts (q)</td>
<td>(p \cap q = \emptyset)</td>
</tr>
<tr>
<td>(p) and (q)</td>
<td>(p \cap q)</td>
</tr>
<tr>
<td>(p) or (q)</td>
<td>(p \cup q)</td>
</tr>
<tr>
<td>(p) is possible</td>
<td>(p \neq \emptyset)</td>
</tr>
<tr>
<td>(p) is necessary</td>
<td>(p = W)</td>
</tr>
</tbody>
</table>

• Discussions: the following formulas are problematic. Identify and correct the problems.

(42) a. \(\forall q \in C[q \rightarrow q \subseteq p]\)
    (Every true proposition in \(C\) entails \(p\).)
   b. \((p \subseteq q) \rightarrow \neg q\)
    (If \(p\) entails \(q\), then \(q\) is false.)

3.3 Intensional-izing the theory of types and compositions

(43) Types \(^2\)
   a. Basic types: \(e\) (individuals/entities) and \(t\) (truth values).
   b. Functional types: If \(\sigma\) and \(\tau\) are types, then \(\langle \sigma, \tau \rangle\) is a type.
   c. Intensional types: If \(\sigma\) is a type, then \(\langle s, \sigma \rangle\) is an intensional type.

(44) Domains
   a. \(D_s = W\)
   b. \(D_{\langle \sigma, \tau \rangle} = \{f \mid f : D_\sigma \rightarrow D_\tau\}\) (functions from things of type \(\sigma\) to things of type \(\tau\))
   c. \(D_{\langle s, \tau \rangle} = \{f \mid f : W \rightarrow D_\tau\}\) (functions from possible worlds to things of type \(\tau\))

(45) Intensional Functional Application
    If \(\{\beta, \gamma\}\) is the set of \(\alpha\)’s daughters, \([\beta] \in D_{\langle s, \sigma \rangle, \tau}\), and \([\gamma] \in D_\sigma\), then \([\alpha] = ([\beta](\lambda w.[\gamma]^w))\)

\(^2\)Note that we are not actually adding \(s\) for possible worlds to our type theory. This is because (as far as we’ve seen) there are no expressions of natural language that have specific possible worlds as their values.