Exhaustivity - George 2013, Klinedinst & Rothschild 2011

Gunnar Lund

November 4, 2016

1 George 2013

1.1 Framework

- George starts with a lambda-abstract theory of questions. That is, he takes (a crucial part of) the meaning of a question to be a lambda abstract, specifically the intension of a lambda abstract, combined with a question formation operator Q.
- For example, the intension of the lambda abstract in (1a) is given in (1b).

  (1) a. Who does Anne love?
  b. $\lambda w \lambda x_e(person(w)(x) \wedge loves(w)(x)(Anne))$

- The denotation of *who*, then, will be the following:

  (2) $who = \lambda S_{\langle \mathtt{e},\mathtt{t}\rangle}\ \lambda x_e(person(w)(x) \wedge S(x))$

- He apparently has a way of solving one of the major problems of this view, type mismatch in multiple-'wh' questions, but he doesn't tackle this until later, and it won't matter for our purposes.

1.2 Answer Sets

- George notes that there are two types of answer sets: mention-some answers and strongly exhaustive answers. Of (1a), a mention-some answer will be a proposition providing one entity that Anne loves. A strongly exhaustive answer will name all such entities.
- This gives us the following two answer sets:

  (3) a. $\lambda p_{\langle \mathtt{s},\mathtt{t}\rangle} \exists y_e(p = \lambda w_s(person(w)(y) \wedge loves(w)(y)(Anne)))$
  "Any proposition such that it names some person that Anne loves"
  b. $\lambda p_{\langle \mathtt{s},\mathtt{t}\rangle} \exists S_{\langle \mathtt{e},\mathtt{t}\rangle}(p = \lambda w_s(S = \lambda x_e(person(w)(y) \wedge loves(w)(y)(Anne))))$
  "Any proposition such that it names the set of people that Anne loves"

- Essentially, (3a) checks set membership, (3b) checks set equality.

1.2.1 Mention-some Answers

- Remember that questions are the combination of a question formation operator and the intension of a lambda abstract.
- Using (1b) as our lambda abstract and (3a) as our fully formed question, it seems easy to get the denotation of the Q operator for mention-some questions: take the form in (3a) and treat the (1b) bits as a variable bound by a lambda of type $\langle \mathtt{s,\ et} \rangle$.
- This will work in this case, but it won't work for cases where the lambda abstract part of the question is a different type. This can be solved by making the operator type-flexible:

  (4) **Question formation operator:**
  $Q_\tau = \lambda\alpha_{\langle \mathtt{s},\ \langle \tau, \mathtt{t} \rangle \rangle} \lambda p_{\langle \mathtt{s},\ \mathtt{t} \rangle} \exists \beta_\tau (p = \lambda w'_s(\alpha(w')(\beta)))$
  - where τ is any type.

- Notice that this formula is essentially the same as (3a) where the instance of (1b) has been replaced by α.

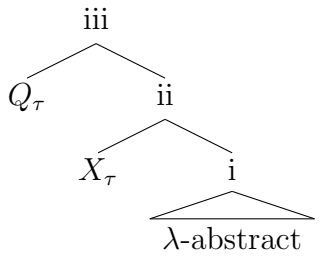
1.2.2 Strongly Exhaustive Answers

- George claims that the form of strongly exhaustive answers is what is given in (3b).
- Instead of simply doing for (3b) what he did with (3a), he introduces a second operator, X, which produces the form in (3b) in tandem with the Q operator.
- X adds exhaustivity. As such, it should identify the extension of some lambda abstract.
- George defines X_τ as follows, for any type τ:

  (5) **Exhaustivity operator:**
  $X_\tau = \lambda\gamma_\tau \lambda\delta_\tau (\gamma = \delta)$

- This is applied to the lambda abstract. The Q operator is then applied to this result, yielding a structure like:

  (6)



- Let's see how this works with "Who does Anne love" with the structure in (6) where node (i) is (1b) $\lambda w \lambda x_e (person(w)(x) \wedge loves(w)(x)(Anne))$:

(7) i: $\lambda w \lambda x_e(person(w)(x) \wedge loves(w)(x)(Anne))$

ii: $X_{\langle \mathtt{e},\mathtt{t}\rangle}(\mathrm{i})$
$= \lambda\gamma_\tau\lambda\delta_\tau(\gamma = \delta)(\lambda w\lambda x_e(person(w)(x) \wedge loves(w)(x)(Anne)))$
$= \lambda\delta_\tau(\lambda x_e(person(w)(x) \wedge loves(w)(x)(Anne)) = \delta)$

$Q_{\langle \mathtt{e},\ \mathtt{t}\rangle}$: $\lambda\alpha_{\langle \mathtt{s},\ \langle \mathtt{e},\ \mathtt{t}\rangle\rangle}\lambda p_{\langle \mathtt{s},\ \mathtt{t}\rangle}\exists\beta_{\langle \mathtt{e},\ \mathtt{t}\rangle}(p = \lambda w'_s(\alpha(w')(\beta)))$

iii: $\lambda p_{\langle \mathtt{s},\ \mathtt{t}\rangle}\exists\beta_{\langle \mathtt{e},\ \mathtt{t}\rangle}(p = \lambda w'_s(\lambda x \begin{pmatrix} person(w')(x)\wedge \\ loves(w')(x)(Anne) \end{pmatrix} = \beta))$

- This is precisely what we want for the strongly exhaustive case.

2 Klinedinst & Rothschild: "Exhaustivity in Questions with Non-Factives"

2.1 Introduction

- Factive and non-factive verbs differ on their requirement of the exhaustivity of embedded questions.
- Namely, there are thought to be two conditions on a true answer:

(8) a. The true answer must be completely specified.
b. The true answer must be specified *as being* the complete answer.

- A weakly exhaustive answer requires only (a). A strongly exhaustive one requires both (a) and (b).
- Typically, non-factive verbs like "tell" and "predict" are weakly exhaustive, while factive verbs like "know" are strongly exhuastive.
- For example, "John predicted who sang" requires John to have predicted only who sang but not who didn't sing. On the other hand, "John knows who sang" requires John to know both who sang and who didn't sing.
- K&R argue that this picture isn't quite right. The requirement is slightly stronger for non-factives than (a), what they call an intermediate reading.
- A non-factive like "predict" requires that, for example, John correctly predicts who sang and also that no one else sang.

2.2 Weak and strong exhaustivity

- Weakly exhaustive readings are mainly thought to pattern with Karttunen answer sets. That is, the denotation of a question is all true answers to the question, à la (9) below for "who sang":

(9) **Weakly Exhuastive:** $[\![\text{who sang}]\!] = \lambda w \lambda w' \forall x (x \text{ sang in } w \rightarrow x \text{ sang in } w')$

- So, someone who knows who sang would know the set of propositions where x sang in w.
- Groenendijk & Stokhof (1984) critique this view as it wrongly predicts that (10b) is true given the context in (10a):

(10) a. Frank and Emilio are the only students who sang. Arthur knows that Frank and Emilio sang, but he also wrongly believes that Bill and Ted sang.
b. Arthur knows who sang.

- The reading in (9) is thus too weak. As such, G&S posit that the following denotation for "who sang" (using partition semantics):

(11) **Strongly Exh.:** $[\![\text{who sang}]\!] = \lambda w \lambda w' \forall x (x \text{ sang in } w \iff x \text{ sang in } w')$

- Problems for this view: it's too strong for non-factives, which allow for statements like the following which would be contradictory with the denotation in (11):

(12) John told me/predicted who sang, but did not tell me/predict who did not sing.

- But this doesn't mean we should use weakly exhaustive reading for non-factives either: If, given a world where only Frank and Emilio sang, it wouldn't be true that "John predicted who sang" if he predicted that Frank, Emilio, *and Ted* sang.
- Having shown the inadequacy of both weak and strong readings for non-factives, K&R argue that intermediate readings get these judgments right.

2.3 Exhaustivity operator

- To get these intermediate readings, K&R propose an exhaustivity operator that acts something like *only*.
- This operator can actually derive both intermediate and strongly exhaustive readings depending on where it is inserted and takes the weakly exhaustive denotation as the "basic" denotation of the question (i.e. what's in (9)).
- This operator requires a *focus value* (similar to, yet quite different from, the standard Roothian view) and what they refer to as $F+$, a further operation on the focus value that takes all propositions stronger than Q:

(13) a. $[\![\text{who came}]\!]_F = \{Q' : \forall w \exists w' (Q'(w) = [\![\text{who came}]\!](w'))\}$
b. $[\![\text{who came}]\!]_{F+} = \{Q' \in [\![\text{who came}]\!]_F : Q' \supset [\![\text{who came}]\!]\}$

- The exhaustivity operator, then, entails the sentence it takes as argument and negates and conjoins all $F+$ values of the sentence:

(14) $[\![\text{EXH } \phi]\!] = [\![\phi]\!] \wedge (\bigwedge_{P \in [\![\phi]\!]_{F+}} \neg P)$

- For non-factives, this operator applies at the top, at the matrix sentence. Take "John predicted who came":

(15) $[\![\text{John predicted who came}]\!] \wedge (\bigwedge_{P \in [\![\text{John predicted who came}]\!]_{F+}} \neg P)$

- So, it entails the weak reading but also entails that John didn't predict something stronger.
- Factives like "know" require a stronger reading than even this. To obtain this stronger reading, K&R argue that the exhaustivity operator should scope directly above the question.
- To see how this works, see EXH applied to "who sang" below:

(16) $[\![\text{EXH [who sang]}]\!]$
$= \lambda w \lambda w' \forall x (x \text{ sang in } w \rightarrow x \text{ sang in } w') \wedge (\bigwedge_{P \in [\![\text{who sang}]\!]_{F+}} \neg P(w)(w'))$

- This is essentially a set of answers that specify not only who sang but also those who didn't sing.
- When embedded under "know", we get the strongly exhaustive reading that we're looking for.
- K&R note that there are some issues with this account, namely the scope of EXH among other operators, the behavior of certain verbs like "foretell", and *de re/de dicto* readings.

3 Summary

- Both papers make use of an operator that "exhausts" possibilities, though they do so in different ways and for different reasons.
- In George 2013, we see the use of an exhaustivity operator that distinguishes mention-some questions from mention-all questions. Namely, the operator appears directly above the lambda abstract in mention-all cases and doesn't appear in mention-some questions.
- In K&R 2011, the operator is used to distinguish weakly exhaustive and strongly exhaustive readings of questions embedded under factive and non-factive verbs. It's the syntactic position of this operator, not its presence, that derives the two different readings.