Composing local contexts

Chris Barker, version of *May* 5, 2021

- Tracking ***incremental semantic commitments*** explains
  - local presupposition satisfaction
  - bounded modality
  - (Anaphora another day)

Two empirical targets

- Local presupposition satisfaction

1. It rained, and Ann knows it rained. [no presup]
2. Ann knows it rained, and it rained. [presup: it rained]

- Bounded modality
  - Mandelkern 2019: the information in one conjunct constrains the epistemic accessibility relation involved in evaluating the other conjunct.

3. It is not raining, and it might be raining.

Local contexts

Karttunen 1974: "In compound sentences, the initial context is incremented in a left-to-right fashion giving for each constituent a ***local context*** that must satisfy its presuppositions."

Implementations:

- Karttunen: contexts are **sets of logical forms**. Add the logical forms of embedded clauses to the initial context
- Heim 1983: cash out contexts as **sets of worlds**, update with intersection (reconceives meanings as CCPs)
- Schlenker 2007: "the only information that needs to be updated concerns ***the words that the speech act participants have pronounced***."
- View today: The only information that needs to be updated concerns ***the semantic commitments of the expressions that have already been evaluated***.

Evalution order

- First approximation: linear order
- Second approximation: linear order, except
  - quantifier scope relations
  - reconstruction
- Today's strategy:
  - assume post-QR logical form encodes the relevant order
  - George 2007 makes this assumption too

Schlenker: quantifying over possible syntactic completions

Given a context set C, a predicative or propositional occurrence of d is infelicitous in a sentence that begins with 'a (d and' if for any expression g of the same type as d and for any sentence completion b, $C \Vdash a(d \text{ and } g)b \Leftrightarrow agb$.

Figure 1: Schlenker 2008's formulation of the maxim "Be Brief!"

- It's not about syntactic completions, it's about meanings
- But how? Continuations! (or something like continuations)

The core technique

- Two type shifters: $\textsc{t}$ (Partee's $\textsc{lift}$) and a new shifter $\textsc{h}$:

$\textsc{t} = \lambda x\kappa.\kappa x$ e.g., $\textsc{t}$ **ann** $= \lambda P.P$ **ann**

$\textsc{h} = \lambda LR\kappa.R(L(\lambda xy.\kappa(xy)))$

Recipe for dynamicizing a standard logical form:

- Use $\textsc{h}$ for function application
- Use $\textsc{t}$ for lexical items that don't trigger presuppositions
- (Predicate abstraction omitted; ask me)

Simple example without presuppositions:

1. a. [Ann slept] Standard logical form
   b. $\textsc{h}$ ($\textsc{t}$ **ann**) ($\textsc{t}$ **slept**) Dynamicized logical form

Continuing the example

$\textsc{t} = \lambda x\kappa.\kappa x$ $\qquad\qquad$ $\textsc{h} = \lambda LR\kappa.R(L(\lambda xy.\kappa(xy)))$

1. a. [Ann slept] — Standard logical form
   b. $\textsc{h}$ ($\textsc{t}$ **ann**) ($\textsc{t}$ **slept**) — Dynamicized logical form
   c. $= (\lambda LR\kappa.R(L(\lambda xy.\kappa(xy))))\ (\textsc{t}\ \mathbf{ann})\ (\textsc{t}\ \mathbf{slept})$
   d. $\rightsquigarrow_\beta \lambda\kappa.(\textsc{t}\ \mathbf{slept})\ ((\textsc{t}\ \mathbf{ann})\ (\lambda xy.\kappa(xy)))$
   e. $\rightsquigarrow_\beta \lambda\kappa.(\textsc{t}\ \mathbf{slept})\ ((\lambda\kappa.\kappa\ \mathbf{ann})\ (\lambda xy.\kappa(xy)))$
   f. $\rightsquigarrow_\beta \lambda\kappa.(\textsc{t}\ \mathbf{slept})\ (\lambda y.\kappa(\mathbf{ann}\ y))$ ***
   g. $\rightsquigarrow_\beta \lambda\kappa.(\lambda\kappa.\kappa\ \mathbf{slept})\ (\lambda y.\kappa(\mathbf{ann}\ y))$
   h. $\rightsquigarrow_\beta \lambda\kappa.(\lambda y.\kappa(\mathbf{ann}\ y))\ \mathbf{slept}$
   i. $\rightsquigarrow_\beta \lambda\kappa.\kappa(\mathbf{ann\ slept})$

- To recover the usual denotation, apply to $\textsc{i} = \lambda x.x$:
  a. $(\lambda\kappa.\kappa(\mathbf{ann\ slept}))\ \textsc{i}$
  b. $\rightsquigarrow_\beta \textsc{i}\ (\mathbf{ann\ slept})$
  c. $= (\lambda x.x)\ (\mathbf{ann\ slept})$
  d. $\rightsquigarrow_\beta \mathbf{ann\ slept}$
- Bottom line: dynamicized LF computes the static denotation

What dynamicization enables

- Here, a ***local context*** κ is a function from a local denotation to the semantic commitments of the expressions that have been evaluated so far
- The system guarantees that each expression takes its local context as its first semantic argument.
- So the denotation of each expression has direct semantic access to its local context.

Defining local presupposition satisfaction

3. a. [It rained [and [Ann [knows it rained]]]]
   b. $\textsc{h}$ ($\textsc{t}$ **rain**)($\textsc{h}$ ($\textsc{t}$ **and**)($\textsc{h}$ ($\textsc{t}$ **ann**) **knows-it-rained**)) $\textsc{i}$
   c. $\rightsquigarrow_\beta$ **knows-it-rained** (λP.**and** (P **ann**) **rain**)

- Local context of *knows it rained*: $\kappa = \lambda P$.**and** (P **ann**) **rain**
- No matter what P turns out to be, κP guarantees it rained.
- Definition of "guaranteed no matter what":

$$\lfloor \kappa \rfloor = \begin{cases} \kappa & \text{if } \kappa \text{ has type } \mathtt{st} \\ \exists x_{\mathtt{a}}.\lfloor \kappa x \rfloor & \text{if } \kappa \text{ has type } \mathtt{a} \to \mathtt{b} \end{cases} \tag{1}$$

- $\lfloor \lambda P.\textbf{and}\ (P\ \textbf{ann})\ \textbf{rain} \rfloor = \exists P.\textbf{and}\ (P\ \textbf{ann})\ \textbf{rain}$
- **Presupposition satisfaction**: the presupposition p of an expression with local context κ is satisfied just in case $\lfloor \kappa \rfloor \to p$.
- Lexical entry for *know*: $\lambda\kappa.\kappa(\lambda p : \lfloor \kappa \rfloor \to p.\textbf{know}\ p)$

Illustration of presupposition failure:

4. a. [[Ann [knows it rained]] [and [it rained]]]
   b. $\textsc{h}$ ($\textsc{h}$ ($\textsc{t}$ **ann**) **knows-it-rained**)($\textsc{h}$ ($\textsc{t}$ **and**)($\textsc{t}$ **rain**)) $\textsc{i}$
   c. $\rightsquigarrow_\beta$ **knows-it-rained** $(\lambda Pf.f(P\ \textbf{ann}))$ (**and rain**)

- Local context of *knows it rained*: $\kappa = \lambda Pf.f(P\ \textbf{ann})$
- $\lfloor\kappa\rfloor \not\rightarrow$ **rain**

Bounded modality (asymmetric version)

- Lexical entry for *might*: $\lambda\kappa.\kappa(\lambda p.\exists w' \in \textsc{dox}_w.[\![\lfloor\kappa\rfloor \wedge p]\!]^{w'})$

5. a. It is not raining and it might be raining.
   b. $\textsc{h}$ ($\textsc{h}$ ($\textsc{t}$ **not**) ($\textsc{t}$ **rain**)) ($\textsc{h}$ ($\textsc{t}$ **and**) ($\textsc{h}$ **might** ($\textsc{t}$ **rain**))) $\textsc{i}$
   c. $\rightsquigarrow_\beta$ **might** ($\lambda mp.$**and** (**not rain**) (mp)) **rain**
   d. $\rightsquigarrow_\beta$ **and** (**not rain**) $(\lambda w.\exists w' \in$
   $\textsc{dox}_w.[\![(\exists p.$**and** (**not rain**) $p) \wedge$ **rain**$]\!]^{w'})$

- Unlike Mandelkern 2019,
  - the local context is part of the truth conditions of *might*
    - could put it into the satt conditions, like Mandelkern
  - local contexts are leftward contexts only, not symmetric
    - see next slide

Symmetric local contexts

- Key facts about $\textsc{h}$:
  - the continuation delivered by $\textsc{h}$ contains all and only the semantic commitments of the expressions that have already been evaluated; and
  - it evaluates expressions from left to right
- In order to have symmetric local contexts, replace $\textsc{h}$ with $\textsc{h}_\textsc{s}$:
- $\textsc{h} = \lambda L R \kappa . R(L(\lambda x y . \kappa(x y)))$
- $\textsc{h}_\textsc{s} = \lambda L R \kappa . L(\lambda x . R(\lambda y . \kappa(x y)))$
- In a $\textsc{h}_\textsc{s}$-based system, the local context κ contains the commitments of the entire surrounding utterance, including expressions that have not yet been evaluated

1. Ann will leave too if Bill leaves.
2. It might be raining but it's not raining.

[Exercise: compute local contexts of *too* and *might* using $\textsc{h}$ and $\textsc{h}_\textsc{s}$]

Conclusions

- The lifted computation is purely semantic, and does not involve quantifying over syntactic completions
- Expressions are evaluated with respect to a single world
  - [*Go team pointwise!*]
- The left-right asymmetry is systematic across all expressions
- In particular, logical connectives receive no special treatment
  - they bear their standard bivalent truth conditional meaning, and
  - they undergo the same simple lifting operation as any other lexical item that doesn't trigger presuppositions
- Furthermore, the left-right asymmetry is located in a single place in the system, namely, in the H combinator.
- Local contexts can be computed symmetrically if desired

Thanks for a great seminar experience!!

Happy counterexamples–