Modularity in semantics: pronouns

Semantics II

April 19, 2018

Modularity

A baseline extensional semantic theory

Inductively define the space of possible meanings, sorted by type:

$$\tau ::= e \mid t \mid \underbrace{\tau \to \tau}_{e\to t,\ (e\to t)\to t,\ \ldots}$$

Interpret binary combination via functional application:

$$[\![\alpha\ \beta]\!] := [\![\alpha]\!]\,[\![\beta]\!]$$

Assignment-dependence

Natural languages have free and bound pro-forms.

1. John saw her.
2. Everybody$_i$ did their$_i$ homework.

Standardly: meanings depend on assignments (ways of valuing free variables).

$$\sigma ::= \mathtt{e} \mid \mathtt{t} \mid \sigma \to \sigma \qquad\qquad \tau ::= \mathtt{G}\sigma ::= \mathtt{g} \to \sigma$$

Interpret binary combination via **assignment-sensitive** functional application.

$$[\![\alpha\ \beta]\!] := \lambda g.\, \underbrace{\underbrace{[\![\alpha]\!]}_{\mathtt{G}(b \to c)}\, g\, (\underbrace{[\![\beta]\!]}_{\mathtt{G}b}\, g)}_{\mathtt{G}c}$$

Sample derivation: *John saw her₀*

- $\mathtt{G\,t}$ — $\lambda g.\,\mathsf{saw}\,g_0\,\mathsf{j}$
  - $\mathtt{G\,e}$ — $\lambda g.\,\mathsf{j}$
  - $\mathtt{G\,(e \to t)}$ — $\lambda g.\,\mathsf{saw}\,g_0$
    - $\mathtt{G\,(e \to e \to t)}$ — $\lambda g.\,\mathsf{saw}$
    - $\mathtt{G\,e}$ — $\lambda g.\,g_0$

(Apply the result to a contextually furnished assignment to get a proposition.)

Pulling out what matters

Key features of the standard approach to assignment-dependence:

- Uniformity: everything depends on an assignment (many things trivially).
- Enriched composition: $[\![\cdot]\!]$ stitches assignment-relative meanings together.

Here's another possibility: abstract out these key pieces, apply them on demand.

$$\underbrace{\rho\, x := \lambda g.\, x}_{\text{cf. } [\![\text{John}]\!] := \lambda g.\mathbf{j}} \qquad\qquad \underbrace{m \circledast n := \lambda g.\, mg\,(ng)}_{\text{cf. } [\![\alpha\ \beta]\!] := \lambda g.[\![\alpha]\!]\,g\,([\![\beta]\!]\,g)}$$

Sample derivation: *she₀ spoke*

$\mathtt{G\,t}$

$\lambda g.\,\text{spoke}\,g_0$

$\mathtt{G\,e}$	$\mathtt{G\,e \to G\,t}$
$\lambda g.\,g_0$	$\lambda n.\lambda g.\,\text{spoke}\,(n\,g)$

$\Big\uparrow \circledast$

$\mathtt{G\,(e \to t)}$

$\lambda g.\,\text{spoke}$

$\Big\uparrow \rho$

$\mathtt{e \to t}$

spoke

Applicatives

G's ρ and $\circledast$ make it an **applicative functor** (McBride & Paterson 2008, Kiselyov 2015). A type constructor F is applicative if it supports ρ and $\circledast$ with these types...

$$\rho : a \to F\,a \qquad\qquad \circledast : F\,(a \to b) \to F\,a \to F\,b$$

...Where ρ is a trivial way to inject something into the richer type characterized by F, and $\circledast$ is function application lifted into F.[1]

[1] To ensure that ρ and $\circledast$ behave as advertised, they'll need to satisfy some laws. These needn't detain us, but see McBride & Paterson 2008, Charlow 2017.

Applicative alternatives

The technique is pretty general. Alternatives follow a similar pattern:[2]

$$\sigma ::= \mathsf{e} \mid \mathsf{t} \mid \sigma \to \sigma \qquad\qquad \tau ::= \mathsf{S}\,\sigma$$

Then S's ρ and $\circledast$ operations are defined as follows:

$$\underbrace{\rho\, x := \{x\}}_{\text{cf. } [\![\text{John}]\!] := \{\mathsf{j}\}} \qquad\qquad \underbrace{m \circledast n := \{f\,x \mid f \in m, x \in n\}}_{\text{cf. } [\![\alpha\ \beta]\!] := \{f\,x \mid f \in [\![\alpha]\!], x \in [\![\beta]\!]\}}$$

[2] As do *continuized* grammars (Shan & Barker 2006, Barker & Shan 2014)!

Sample derivations: *a linguist spoke*/*she$_0$ spoke*

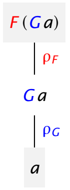
- $\mathtt{S\,t}$ — $\{\text{spoke}\,x \mid x \in \text{ling}\}$
  - $\mathtt{S\,e}$ — $\{x \mid x \in \text{ling}\}$
  - $\mathtt{S\,e \to S\,t}$ — $\lambda n.\{\text{spoke}\,x \mid x \in n\}$
    - $\circledast$
    - $\mathtt{S\,(e \to t)}$ — $\{\text{spoke}\}$
      - ρ
      - $\mathtt{e \to t}$ — spoke

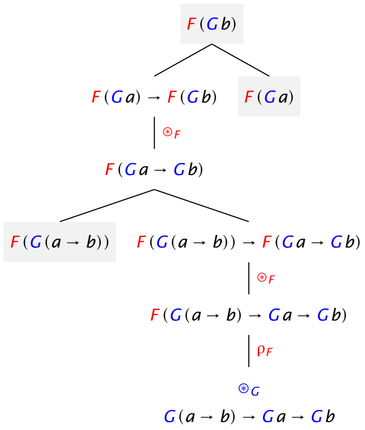
- $\mathtt{G\,t}$ — $\lambda g.\,\text{spoke}\,g_0$
  - $\mathtt{G\,e}$ — $\lambda g.\,g_0$
  - $\mathtt{G\,e \to G\,t}$ — $\lambda n.\lambda g.\,\text{spoke}\,(n\,g)$
    - $\circledast$
    - $\mathtt{G\,(e \to t)}$ — $\lambda g.\,\text{spoke}$
      - ρ
      - $\mathtt{e \to t}$ — spoke

(The applicative abstraction seems like a solid one: it captures a recurrent design pattern.)

Composed applicatives

Whenever F and G are applicative, FG (really, $F \circ G$) is too:





Composed applicatives: examples

There's 2 ways to compose assignment-dependence and alternatives. First, for GS:

$$\rho\, x \coloneqq \lambda g.\, \{x\} \qquad\qquad m \circledast n \coloneqq \lambda g.\, \{f\, x \mid f \in m\, g, x \in n\, g\}$$

This corresponds to standard alternative semantics. And here, for SG:

$$\rho\, x \coloneqq \{\lambda g.\, x\} \qquad\qquad m \circledast n \coloneqq \lambda g.\, \{f\, g\, (x\, g) \mid f \in m, x \in n\}$$

This layering is less common, but appears in e.g. Romero & Novel (2013).

Variable-free semantics

Pronouns as identity maps

Jacobson (1999) proposes we stop thinking of pronouns as assignment-relative and index-oriented. Instead, she suggests we model pronouns as **identity functions**:

$$\mathsf{E}\,a ::= \mathsf{e} \to a \qquad\qquad \textbf{she} := \underbrace{\lambda x.\, x}_{\mathsf{E}\,\mathsf{e}}$$

How should these compose with things like transitive verbs, which are looking for an individual, not a function from individuals to individuals?

Pronouns as identity maps

Jacobson (1999) proposes we stop thinking of pronouns as assignment-relative and index-oriented. Instead, she suggests we model pronouns as **identity functions**:

$$\mathsf{E}\,a ::= \mathsf{e} \to a \qquad\qquad \textbf{she} := \underbrace{\lambda x.\, x}_{\mathsf{E}\,\mathsf{e}}$$

How should these compose with things like transitive verbs, which are looking for an individual, not a function from individuals to individuals?

Of course, this is *exactly* the same problem that comes up when you introduce assignment-dependent meanings! And hence it admits the exact same solution.

Pronominal composition, with and without assignments

- $\lambda g.\,\mathbf{left}\,g_0$ — $\mathtt{G\,t}$
  - $\lambda g.\,g_0$ — $\mathtt{G\,e}$ — she$_0$
  - $\lambda n.\lambda g.\,\mathbf{left}\,(n\,g)$ — $\mathtt{G\,e} \to \mathtt{G\,t}$
    - $\circledast$
      - $\lambda g.\,\mathbf{left}$ — $\mathtt{G}\,(\mathtt{e} \to \mathtt{t})$
        - ρ
          - $\mathbf{left}$ — $\mathtt{e} \to \mathtt{t}$ — left

- $\lambda x.\,\mathbf{left}\,x$ — $\mathtt{E\,t}$
  - $\lambda x.\,x$ — $\mathtt{E\,e}$ — she
  - $\lambda n.\lambda x.\,\mathbf{left}\,(n\,x)$ — $\mathtt{E\,e} \to \mathtt{E\,t}$
    - $\circledast$
      - $\lambda x.\,\mathbf{left}$ — $\mathtt{E}\,(\mathtt{e} \to \mathtt{t})$
        - ρ
          - $\mathbf{left}$ — $\mathtt{e} \to \mathtt{t}$ — left

In an important sense, then, the compositional apparatus underwriting variable-free composition is equivalent to that underwriting assignment-friendly composition!

Heim & Kratzer (1998: 92): $[\![t]\!] = \lambda x.x$

(5) Preliminary definition: An *assignment* is an individual (that is, an element of D (= D_e)).

A trace under a given assignment denotes the individual that constitutes that assignment; for example:

(6) The denotation of "t" under the assignment Texas is Texas.

An appropriate notation to abbreviate such statements needs to be a little more elaborate than the simple $[\![\ldots]\!]$ brackets we have used up to now. We will indicate the assignment as a superscript on the brackets; for instance, (7) will abbreviate (6):

(7) $[\![t]\!]^{\text{Texas}}$ = Texas.

The general convention for reading this notation is as follows: Read "$[\![\alpha]\!]^a$" as "the denotation of α under a" (where α is a tree and a is an assignment).

(7) exemplifies a special case of a general rule for the interpretation of traces, which we can formulate as follows:

(8) If α is a trace, then, for any assignment a, $[\![\alpha]\!]^a = a$.

Multiple pronouns

There is an important difference between using assignments and individuals as reference-fixing devices. Assignments are data structures that can in principle value *every* free pronoun you need. But an individual can only value *co-valued* pronouns!

3. She saw her.

So a variable-free treatment of cases like these must give something like this:

$$\underbrace{\lambda x.\lambda y.\,\mathbf{saw}\,y\,x}_{\mathsf{E(Et)}}$$

Can we derive something of this type, using our existing apparatus?

Composing E with E

You bet we can. Remember that, given two applicative functors F and G, their composition FG is automatically applicative as well. If F and G are both E, we have:

$$\rho\, z =$$

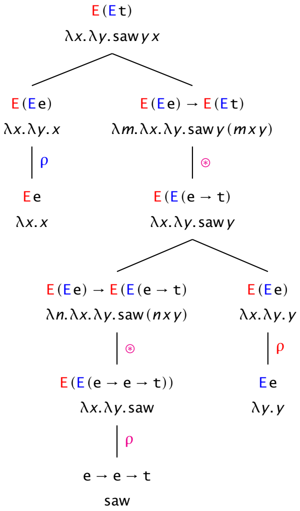
Composing E with E

You bet we can. Remember that, given two applicative functors F and G, their composition FG is automatically applicative as well. If F and G are both E, we have:

$$\rho\, z = \lambda x.\lambda y.\, z \qquad\qquad m \circledast n = \lambda x.\lambda y.\, m\, x\, y\, (n\, x\, y)$$

We are just managing two individuals (tiny assignments) rather than one!

A derivation



Assignments, and "variables", on demand

Witness the curry/uncurry isomorphisms:

$$\textbf{curry}\, f := \lambda x.\lambda y.f\,(x,y) \qquad \textbf{uncurry}\, f := \lambda(x,y).f\,x\,y$$

In other words, by (iteratively) uncurrying a variable-free proposition, you end up with a dependence on a sequence (tuple) of things. Essentially, an assignment.

$$\textbf{uncurry}\,(\lambda x.\lambda y.\textbf{saw}\,y\,x) = \lambda(x,y).\textbf{saw}\,y\,x = \lambda p.\textbf{saw}\,p_1\,p_0$$

Obversely, by iteratively currying a sequence(/tuple)-dependent proposition, you end up with a higher-order function. Essentially, a variable-free meaning.

$$\textbf{curry}\,(\lambda p.\textbf{saw}\,p_1\,p_0) = \textbf{curry}\,(\lambda(x,y).\textbf{saw}\,y\,x) = \lambda x.\lambda y.\textbf{saw}\,y\,x$$

Variable-free semantics?

So variable-free semantics (can) have the same combinatorics as the variable-full semantics. This is no great surprise: they're both about compositionally dealing with "incomplete" meanings.

Moreover, under the curry/uncurry isomorphisms, a variable-free proposition is equivalent to (something) like an assignment dependent proposition.

Let's call the whole thing off?

Barker, Chris & Chung-chieh Shan. 2014. *Continuations and natural language*. Oxford: Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199575015.001.0001.

Charlow, Simon. 2017. A modular theory of pronouns and binding. In *Proceedings of Logic and Engineering of Natural Language Semantics 14*.

Heim, Irene & Angelika Kratzer. 1998. *Semantics in generative grammar*. Oxford: Blackwell.

Jacobson, Pauline. 1999. Towards a variable-free semantics. *Linguistics and Philosophy* 22(2). 117–184. https://doi.org/10.1023/A:1005464228727.

Kiselyov, Oleg. 2015. Applicative abstract categorial grammars. In Makoto Kanazawa, Lawrence S. Moss & Valeria de Paiva (eds.), *NLCS'15. Third workshop on natural language and computer science*, vol. 32 (EPiC Series), 29–38.

McBride, Conor & Ross Paterson. 2008. Applicative programming with effects. *Journal of Functional Programming* 18(1). 1–13. https://doi.org/10.1017/S0956796807006326.

Romero, Maribel & Marc Novel. 2013. Variable binding and sets of alternatives. In Anamaria Fălăuș (ed.), *Alternatives in Semantics*, chap. 7, 174–208. London: Palgrave Macmillan UK. https://doi.org/10.1057/9781137317247_7.

Shan, Chung-chieh & Chris Barker. 2006. Explaining crossover and superiority as left-to-right evaluation. *Linguistics and Philosophy* 29(1). 91–134. https://doi.org/10.1007/s10988-005-6580-7.