Variable-free statics (and some dynamics)

Computational semantics seminar

March 8, 2017

Our standard treatment of variables and binding

Basics

Pronouns have semantic values relative to an assignment function:

$$[\![\text{she}_i]\!]^g := g_i$$

Function application is enriched to be assignment-friendly:

$$[\![A\ B]\!]^g := [\![A]\!]^g [\![B]\!]^g$$

Binding

Binding is achieved via assignment function modification:

$$[\![\lambda_i\ A]\!]^g := \lambda x.[\![A]\!]^{g[i \to x]}$$

An example (part of a larger derivation of *every boy$_i$ likes his$_i$ mom*):

$$[\![\lambda_1\ [t_1\ \text{likes}\ t_1\text{'s mom}]]\!]^g = \lambda x.\mathbf{likes}\,(\mathbf{mom}\,x)\,x$$

Weird?

Where are indices? Not overt in any spoken language. (See Schlenker 2011, Kuhn 2015 for discussion of the interesting case of sign language.)

Might think it's strange to have referential intentions involving variables/assignments?

Not categorematic

$[\![\lambda_i\ A]\!]^g$ *can't* be stated in terms of the meanings of $[\![\lambda_i]\!]^g$ and $[\![A]\!]^g$.

Why?

Not categorematic

$[\![\lambda_i\ A]\!]^g$ *can't* be stated in terms of the meanings of $[\![\lambda_i]\!]^g$ and $[\![A]\!]^g$.

Why? Because we're not interested in the meaning of $[\![A]\!]^g$ at all!!

So abstraction requires a **syncategorematic** rule: λ_i has no meaning per se. Its only purpose is to trigger a non-default mode of composition, imbued with a non-default way of handling assignments.

What's the type of an assignment?

Assignments need to be able to harbor things of arbitrary types, not just simple individuals:

1. Eating donuts$_i$, I like t_i.

So what's the type of an assignment function?

What's the type of an assignment?

Assignments need to be able to harbor things of arbitrary types, not just simple individuals:

1. Eating donuts$_i$, I like t_i.

So what's the type of an assignment function? Not an easy question to answer! In effect, you need a (probably infinite) sequence of assignments, one for each possible type.

Binding without c-command

Does binding succeed here?

1. [Her$_i$ naughtiest student]$_j$, every teacher$_i$ punished t_j.

Binding without c-command

Does binding succeed here?

1. [Her$_i$ naughtiest student]$_j$, every teacher$_i$ punished t_j.

It does not!

[her$_1$ naughtiest student] [λ_2 [every teacher [λ_1 [t_1 punished t_2]]]]

$\rightsquigarrow \forall x \in$ **teacher** : **punished** (**the.naughtiest.student.of** g_1) x

LF c-command is required for binding. To get this, you'd need to QR *every teacher* over *her*$_1$ (which should trigger a WCO violation).

- Instead, we'd like to say something like "the topicalized DP reconstructs into its base position, where its pronoun gets bound".

Coindexing weirdness

Al saw his mom before Bo did can't mean that Al saw Frank's mom before Bo saw Bo's mom.

But indices allow us to make an LF for this reading that asserts a weird kind of identity between the antecedent and elided VPs (Heim 1997):

1. Al saw his$_1$ mom before Bo [λ_1 [t_1 did (see his$_1$ mom)]].

This seems deeply troubling. Indices have a weird quasi-representational status, and it's very tempting to state grammatical constraints (like ellipsis licensing) in terms of relationships between indices. But such cases suggest that doing so is fundamentally misguided.

Paychecks

Pronouns, too, can receive "sloppy" readings:

1. Though John$_i$ likes his$_i$ mom, Bill$_j$ HATES her$_?$.

In principle, the sloppiness can depend on any number of pronouns!

2. The woman$_i$ who made Sears$_j$ believe the bill they$_j$ sent her$_i$ was in the mail was wiser than the woman$_k$ who made Filene's$_l$ believe that it$_?$ hadn't been mailed yet. (after Jacobson 2000: 132)

Are these cases generated in the standard approach?

Paychecks

Pronouns, too, can receive "sloppy" readings:

1. Though John$_i$ likes his$_i$ mom, Bill$_j$ HATES her$_?$.

In principle, the sloppiness can depend on any number of pronouns!

2. The woman$_i$ who made Sears$_j$ believe the bill they$_j$ sent her$_i$ was in the mail was wiser than the woman$_k$ who made Filene's$_l$ believe that it$_?$ hadn't been mailed yet. (after Jacobson 2000: 132)

Are these cases generated in the standard approach? They are not: a pronoun simply means what it means at an assignment.

And of course donkey anaphora

The standard view of binding is inherently *static*: meanings are determined relative to assignments, but assignments aren't part of the *information* carried by a sentence.

1. A linguist$_i$ left the party. She$_i$ was tired.
2. Every farmer who owns a donkey$_i$ beats it$_i$.

Going dynamic means an even more robust role for assignments.

Assignments made monadic

Our type constructor:

$$\mathsf{G}\,a ::= g \to a$$

Along with the corresponding η and $\gg=$ operations:

$$\begin{array}{lll} \eta & :: a \to \mathsf{G}\,a & \gg= \quad :: \mathsf{G}\,a \to (a \to \mathsf{G}\,b) \to \mathsf{G}\,b \\ \eta\,x := \lambda g.x & & m \gg= f := \lambda g.f\,(m\,g)\,g \end{array}$$

Properties of a monad

Associativity of $\gg=$ guarantees island-insensitivity. What does this amount to for pronouns?

The availability of higher-order meanings, type $\mathsf{G}(\mathsf{G}\,a)$, guarantees selectivity outside islands. What does this amount to for pronouns?

Variable-free semantics

Overview and main goals

Jacobson (1999) would like a better theory of binding, one which does away with things like indices and assignments.

Why? Well, we've already seen some empirical trouble for the standard picture. It'd be nice to get a better handle on some of that stuff.

Jacobson has bigger fish to fry though: she's interested in a **Directly Compositional** view of the syntax-semantics interface.

Direct Compositionality

Meanings are assigned at surface structure. There is no LF, and every constituent is associated with a meaning.

Logically speaking, this is independent of the question of whether we have indices and assignments.

But people tend to like stating grammatical constraints in terms of representational properties of indices (think: Conditions ABC). These constraints aren't compositional, and Jacobson would like to junk em.

Pronoun meanings

Jacobson proposes we stop thinking of pronouns as assignment-relative and index-oriented.

Instead, she suggests we model pronouns as simple **identity functions** from individuals to individuals (possibly partial ones):

$$\textbf{she} :: e \to e$$
$$\textbf{she} := \lambda x.x$$

So we replace g-relative meanings with e-relative meanings!

A compositional challenge

Not gonna compose with a VP or a transitive verb!

met :: e → e → t **her** :: e → e

So some magic needs to happen. How should that go?

Introducing **G**

Composition with pronouns is facilitated by a type-shifter **G**:

$$\mathbf{G} :: (a \to b) \to (c \to a) \to c \to b$$
$$\mathbf{G} = \lambda f.\lambda g.\lambda x.f\,(g\,x)$$

Introducing **G**

Composition with pronouns is facilitated by a type-shifter **G**:

$$\mathbf{G} :: (a \to b) \to (c \to a) \to c \to b$$
$$\mathbf{G} = \lambda f.\lambda g.\lambda x.f\,(g\,x)$$

Applied to **met**, composing with **her**:

$$\begin{aligned}\mathbf{G\,met\,her} &= \lambda x.\mathbf{met}\,(\mathbf{her}\,x)\\ &= \lambda x.\mathbf{met}\,x\\ &= \mathbf{met}\end{aligned}$$

Introducing **G**

Composition with pronouns is facilitated by a type-shifter **G**:

$$\mathbf{G} :: (a \to b) \to (c \to a) \to c \to b$$
$$\mathbf{G} = \lambda f.\lambda g.\lambda x.f\,(g\,x)$$

Applied to **met**, composing with **her**:

$$\begin{aligned}\mathbf{G}\,\mathbf{met}\,\mathbf{her} &= \lambda x.\mathbf{met}\,(\mathbf{her}\,x)\\ &= \lambda x.\mathbf{met}\,x\\ &= \mathbf{met}\end{aligned}$$

G is just (unary) **function composition!**

$$\mathbf{G}\,f\,g = f \circ g = \lambda x.f\,(g\,x)$$

Is this enough?

Well not quite: it only allows pronouns inside arguments!

1. John $\underbrace{\text{saw her}}$.
   $\lambda x.\lambda y.\mathbf{saw}\,x\,y :: \mathtt{e}\to\mathtt{e}\to\mathtt{t}$

Solution: add another operation to turn *John* into a function.

$$\mathbf{L} :: a \to (a \to b) \to b$$
$$\mathbf{L} = \lambda x.\lambda f.f\,x$$

And now you can derive a meaning for the sentence:

$$\begin{aligned}\mathbf{G}\,(\mathbf{Lj})\,(\mathbf{G}\,\mathbf{met}\,\mathbf{her}) &= \mathbf{Lj} \circ \mathbf{met} \circ \mathbf{her} \\ &= \lambda x.\,\mathbf{met}\,x\,\mathbf{j}\end{aligned}$$

The syntax of **G**

Of course, *John met her* can't mean $\lambda x.\,\mathbf{metj}\,x$! So the **G** rule needs to make type-level distinctions a bit finer-grained than what we've seen:

$$\mathbf{G} :: (a \to b) \to (c \rightsquigarrow a) \to c \rightsquigarrow b$$

We'll ignore this complication today.

Introducing binding

So that's how free pronouns work. How about bound readings in constructions like *every man likes his mother*?

For these cases, Jacobson introduces another type-shifter that's complementary to **G**:

$$\mathbf{Z} :: (a \to b \to c) \to (b \to a) \to b \to c$$
$$\mathbf{Z} = \lambda f.\lambda g.\lambda x.f\,(g\,x)\,x$$

An example for *likes his mother*:

$$\begin{aligned}\mathbf{Z}\,\mathbf{likes}\,(\mathbf{G}\,\mathbf{mom}\,\mathbf{his}) &= \lambda x.\mathbf{likes}\,(\mathbf{G}\,\mathbf{mom}\,\mathbf{his}\,x)\,x\\ &= \lambda x.\mathbf{likes}\,(\mathbf{mom}\,x)\,x\end{aligned}$$

A free paycheck?

Applying **G** to a pronoun has an interesting effect:

$$\begin{aligned}\mathbf{G\,her} &= (\lambda f.\lambda g.\lambda x.f\,(g\,x))\,(\lambda y.y)\\ &= \lambda g.\lambda x.(\lambda y.y)\,(g\,x)\\ &= \lambda g.\lambda x.g\,x \qquad\qquad \text{type: } (\mathsf{e}\to\mathsf{e})\to\mathsf{e}\to\mathsf{e}\end{aligned}$$

Let's see how this combines with a **Z**-ed verb:

$$\begin{aligned}\mathbf{G}\,(\mathbf{Z\,likes})\,(\mathbf{G\,her}) &= \lambda g.\mathbf{Z\,likes}\,(\lambda x.g\,x)\\ &= \lambda g.\mathbf{likes}\,(g\,x)\,x\end{aligned}$$

Folding in the subject, we have $\lambda g.\mathbf{likes}\,(g\,\mathbf{j})\,\mathbf{j}$.

Complicated derivations

Variable-free derivations are rather complex and hard to construct. This only multiplies in more complicated examples.

1. Every man thinks that every boy said that his mother loves his dog.

How would you compose up the embedded clause? How would you ensure that each binder was linked to the correct pronoun?

Complicated coordinations

Binding is possible in examples like the following:

1. Every boy$_i$ told every girl$_j$ that [he$_i$ likes her$_j$] and [she$_j$ likes him$_i$].

How will Jacobson derive such cases?

Scoping the pronoun

Imagine a derivation like the following:

$$\mathbf{G}\,(\lambda x.\,\mathbf{met}\,x\,\mathbf{j})\,\mathbf{her} = \lambda x.\,\mathbf{met}\,x\,\mathbf{j}$$

So scoping the pronoun allows derivations to be simplified! And sometimes, super-simplified:

$$\mathbf{G}\,(\lambda y.\,\mathbf{G}\,(\lambda x.\,\mathbf{met}\,x\,y)\,\mathbf{her})\,\mathbf{she} = \lambda y.\,\lambda x.\,\mathbf{met}\,x\,y$$

And you could even do this with clauses as a whole!

$$\mathbf{G}\,(\lambda p.\,\mathbf{thinks}\,p\,\mathbf{j})\,(\lambda x.\,\mathbf{left}\,x) = \lambda x.\,\mathbf{thinks}\,(\mathbf{left}\,x)\,\mathbf{j}$$

Does this remind you of anything?

Binding out of DP?

Quantifiers can bind pronouns even when the quantifier isn't an argument of the verb:

1. A member of every committee$_i$ voted to abolish it$_i$.

Jacobson relies on **Z**-ing the verb to effect binding. It's not obvious how that should extend to such cases. See Barker (2005) for in-depth discussion.

Shan & Barker (2006)

Shan & Barker (2006) treat pronouns as things that take scope:

$$\textbf{she} := \lambda f.\lambda x.f\,x$$

In general, for this kind of account to work, pronouns must take scope immediately under their binders. This is a somewhat uncomfortable fit, given that pronouns can be arbitarily far away from their binders!

Monadic VFS

What is **G**?

It's **fmap** for the Reader monad (the monad for assignment-dependence)!

$$m \gg\!= \lambda x.\, \eta\,(f\,x) = \lambda g.\, f\,(m\,g) = f \circ m = \mathbf{G}\,f\,m$$

What is **Z**?

Repeating our definition of **Z**.

$$\mathbf{Z} :: (a \to b \to c) \to (b \to a) \to b \to c$$
$$\mathbf{Z} = \lambda f.\lambda g.\lambda x.f\,(g\,x)\,x$$

Does this remind you of anything?

What is **Z**?

Repeating our definition of **Z**.

$$\mathbf{Z} :: (a \to b \to c) \to (b \to a) \to b \to c$$
$$\mathbf{Z} = \lambda f.\lambda g.\lambda x.f\,(g\,x)\,x$$

Does this remind you of anything? It's a slightly shuffled version of the Reader monad's $\gg\!=$!

$$(\gg\!=) :: (\mathsf{g} \to a) \to (a \to \mathsf{g} \to c) \to \mathsf{g} \to c$$
$$m \gg\!= f := \lambda g.f\,(m\,g)\,g$$

So VFS is a demi-monad

It has analogs of **fmap** and $\gg=$, but no η (or μ).

VFS as a monad

Our type constructor:

$$\mathsf{R}_\sigma\, a ::= \sigma \to a$$

Assignment-dependence replaced with individual-dependence:

$$\begin{array}{lll} \eta & :: a \to \mathsf{R}_\sigma\, a & \gg\!= \quad :: \mathsf{R}_\sigma\, a \to (a \to \mathsf{R}_\sigma\, b) \to \mathsf{R}_\sigma\, b \\ \eta\, x := \lambda y.\, x & & m \gg\!= f := \lambda x.\, f\, (m\, x)\, x \end{array}$$

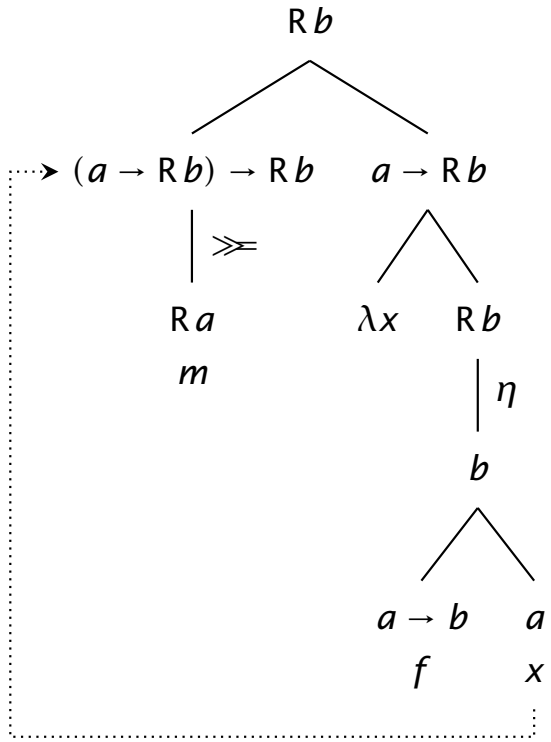
Most often, we'll be interested in $\sigma ::= \mathsf{e}$, in which case we'll just hide the subscript on the type constructor.

Reconstructing Jacobson's shifters

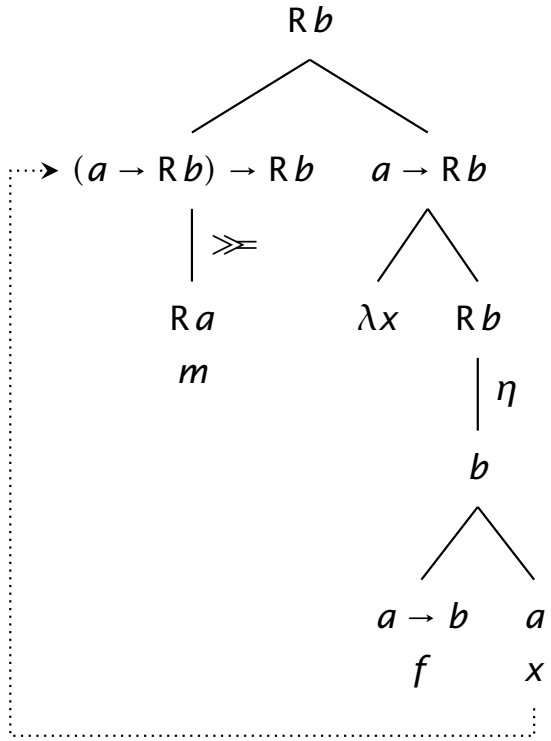
Instead of **Z**-ing a function, we $\gg=$ (flip-**Z**) pronouns and any constituents with unbound pronouns inside them.

How about **G**? It's just the R_{σ} monad's `fmap` operation (every monad is an applicative, and every applicative is a functor), so it's *derivable*!

G regained



G regained



$$= \lambda x.f\,(m\,x) = f \circ m = \mathbf{G}\,f\,m$$

Improved coverage

1. Every boy$_i$ told every girl$_j$ that [he$_i$ likes her$_j$] and [she$_j$ likes him$_i$].
2. Binding out of DP

The curry-uncurry isomorphism(s)

$$\begin{aligned}\mathbf{curry} &:: ((a \times b) \to c) \to a \to b \to c\\ \mathbf{curry}\, f &:= \lambda x.\lambda y.f\,(x, y)\end{aligned}$$

$$\begin{aligned}\mathbf{uncurry} &:: (a \to b \to c) \to (a \times b) \to c\\ \mathbf{uncurry}\, f &:= \lambda (x, y).f\,x\,y\end{aligned}$$

Both of these mappings are isomorphisms:

- $\mathbf{curry} \circ \mathbf{uncurry} = \mathbf{id}_{(a\to b\to c)\to a\to b\to c}$
- $\mathbf{uncurry} \circ \mathbf{curry} = \mathbf{id}_{((a\times b)\to c)\to (a\times b)\to c}$

$$\frac{\mathrm{R}_{(r,s)}\,a}{\mathrm{R}_r\,(\mathrm{R}_s\,a)} = \frac{(r, s) \to a}{r \to s \to a} = \frac{a^{r\cdot s}}{(a^s)^r} = 1$$

Going dynamic

The State monad

A type constructor:

$$\mathsf{ST}\,a ::= \mathsf{g} \to (a, \mathsf{g})$$

Then find a η and a $\gg=$ that make sense:

The State monad

A type constructor:

$$\mathsf{ST}\,a ::= \mathsf{g} \to (a, \mathsf{g})$$

Then find a η and a $\gg\!=$ that make sense:

$$\eta \quad :: \; a \to \mathsf{ST}\,a \qquad\qquad \gg\!= \quad :: \; \mathsf{ST}\,a \to (a \to \mathsf{ST}\,b) \to \mathsf{ST}\,b$$
$$\eta\,x := \lambda g.(x, g) \qquad\qquad m \gg\!= f := \lambda g.f\,(m\,g)_{fst}\,(m\,g)_{snd}$$

Can it be generalized?

Of course! Here's a type constructor:

$$\mathsf{ST}_\sigma\, a ::= \sigma \to (a, \sigma)$$

Then find a η and a $\gg=$ that make sense:

Can it be generalized?

Of course! Here's a type constructor:

$$\mathsf{ST}_\sigma\, a ::= \sigma \to (a, \sigma)$$

Then find a η and a $\gg\!=$ that make sense:

$$\eta \quad :: a \to \mathsf{ST}_\sigma\, a \qquad\qquad \gg\!= \quad :: \mathsf{ST}_\sigma\, a \to (a \to \mathsf{ST}_\sigma\, b) \to \mathsf{ST}_\sigma\, b$$
$$\eta\, x := \lambda y.(x, y) \qquad\qquad m \gg\!= f := \lambda x.f\,(m\,x)_{fst}\,(m\,x)_{snd}$$

What does dref introduction look like?

$$\beta \quad :: \ (a \to \mathsf{ST}_\sigma\, b) \to a \to \mathsf{ST}_\sigma\, b$$
$$\beta\, f\, x := \lambda\_.\, f\, x\, x$$

References I

Barker, Chris. 2005. Remark on Jacobson 1999: Crossover as a local constraint. *Linguistics and Philosophy* 28(4). 447–472. http://dx.doi.org/10.1007/s10988-004-5327-1.

Heim, Irene. 1997. Predicates or formulas? Evidence from ellipsis. In Aaron Lawson (ed.), *Proceedings of Semantics and Linguistic Theory 7*, 197–221. Ithaca, NY: Cornell University. http://dx.doi.org/10.3765/salt.v7i0.2793.

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

Jacobson, Pauline. 2000. Paycheck pronouns, Bach-Peters sentences, and variable-free semantics. *Natural Language Semantics* 8(2). 77–155. http://dx.doi.org/10.1023/A:1026517717879.

Kuhn, Jeremy. 2015. *Cross-categorial singular and plural reference in sign language*. New York University Ph.D. thesis.

Schlenker, Philippe. 2011. Donkey anaphora: the view from sign language (ASL and LSF). *Linguistics and Philosophy* 34(4). 341–395. http://dx.doi.org/10.1007/s10988-011-9098-1.

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