November 21: Dynamic semantics

1 Discourse referents

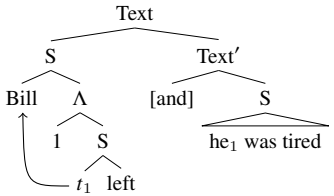
An idea we explored in the previous class: discourse referent ('dref') introduction means assignment function modification.

More concretely: by the time we get around to processing the second sentence of (1), the assignment function we use has changed! In particular, it will map 1 to the senator who admires Kennedy.

(1) [Only one senator]$_1$ admires Kennedy. He$_1$'s very junior.

Today, we'll work up to a theory of how this might work. It'll involve some substantial departures from how we've been thinking of meaning, though the compositional machinery will stay pretty conservative.

In fact, assignment shifting is *already* a part of our theory. In the LF below, t_1 ends up evaluated at an assignment that maps 1 to Bill, i.e. $g^{[b/1]}$.



However: the way we modify assignments doesn't allow *he*$_1$ to get evaluated at this shifted assignment too. Assignment shifting is tied too closely to the presence of a c-commanding (at LF!) abstraction index.

What we'll do today is formulate a theory that can do this. Along the way, we will arrive at an arguably more compositional treatment of variables than our "official" theory managed.

You can think of this as one way of making precise what H&K are doing in Chapter 11 (in particular, giving us a handle on *about-ness*), with the additional benefit that it will allow a uniform account of E-type and garden-variety (i.e. free and bound) pronouns.

2 Meanings as assignment shifters

Our departure point: discourse referent introduction corresponds to assignment function modification.

One way to cash this out (there might be others): the meaning of a sentence is itself the sort of thing that can update an assignment function.

On this perspective, sentence meanings map assignment functions into *new* assignment functions. That is, their type is $\langle a, a\rangle$. Notice that this implies a new role for assignment functions in the theory.

So e.g. the meaning of *Bill*$_n$ *left* is a device for taking an assignment and updating it:

$$[\![\text{Bill}_n \text{ left}]\!] = \lambda g.g^{[b/n]}$$

Another way to write this, more iconically. Those of you who've read Heim (1983) on presupposition projection have seen something similar (Heim uses + instead of brackets):

$$g[\text{Bill}_n \text{ left}] = g^{[b/n]}$$

If sentence meanings are devices for updating assignment functions, the job of *and* must be to compose two updates—that is, to create an assignment function pipeline from the input to the first conjunct to the second conjunct to the output!

$$[\![\text{S}_1 \text{ and } \text{S}_2]\!] = \lambda g.[\![\text{S}_2]\!]([\![\text{S}_1]\!](g))$$

And in the more iconic notation:

$$g[\text{S}_1 \text{ and } \text{S}_2] = g[\text{S}_1][\text{S}_2]$$

In principle, then, the assignment modified by S_1 is available for the evaluation of S_2. S_1 shifts the context of evaluation for S_2.

Incorporating a notion of *order*! This notion of conjunction is not a commutative one. The S_1-modified assignments are fed to S_2, and not vice versa (that is, S_1 is evaluated at the input assignment g).

3 Formally

This is the basic picture: meaning as **context change potential** (where we are working with a somewhat denuded notion of what a context is).

In reality: inadequate. What happened to truth conditions? We would ideally like a way to distinguish *true* sentences from false ones. But the functional $\langle a, a\rangle$ perspective doesn't allow this. The only options are apparently success or undefinedness.[1]

[1] There are ways around this difficulty, in fact, but they have not been explored very much in the linguistics literature. If you're interested, I take this up in the second chapter of my dissertation. ;)

Solution: sentence meanings are functions into *sets* of assignments: $\pi ::= \langle a, \langle a, t\rangle\rangle$. To be false is to return the empty set!

Here is a representative example. All we have done is wrap set braces around the previous output assignment function:

$$[\![\text{Bill}_n \text{ left}]\!] = \lambda g. \begin{cases} \{g^{[\text{b}/n]}\} \text{ if } \textit{left}'(\text{b}) \\ \text{else } \emptyset \end{cases}$$

Equivalently: the relational view (recall from earlier in the term that relations are just functions into sets; e.g. $\textit{likes}'(x)$ is [the characteristic function of] the set of individuals who like x).

$$[\![\text{Bill}_n \text{ left}]\!] = \{\langle g, h\rangle : \textit{left}'(\text{b}) \text{ and } h = g^{[\text{b}/n]}\}$$

More iconically:

$$g[\text{Bill}_n \text{ left}]h \text{ iff } \textit{left}'(\text{b}), \text{ and } h = g^{[\text{b}/n]}$$

If Bill *didn't* leave, there will be no h such that $g[\text{Bill}_n \text{ left}]h$.

So, in sum, we've enriched the perspective a little bit. Instead of just talking about truth relative to an assignment, we have a notion of meaning that allows us to talk about truth relative to an assignment (non-emptiness), as well as assignment change.

Pronouns work as you'd expect.

$$[\![\text{he}_n \text{ was tired}]\!] = \lambda g. \begin{cases} \{g\} \text{ if } \textit{tired}'(g(n)) \\ \text{else } \emptyset \end{cases}$$

4 Lexical semantics

4.1 Verbs

Intransitives like *left* or *was tired* can be modeled as functions from individuals into propositions, i.e. with type $\langle e, \pi\rangle$:

$$[\![\text{left}]\!] = \lambda x. \lambda g. \begin{cases} \{g\} \text{ if } \textit{left}'(x) \\ \text{else } \emptyset \end{cases}$$

Transitives like *likes* and *licked* can be modeled as functions from two individuals into propositions, i.e. with type $\langle e, \langle e, \pi\rangle\rangle$:

$$[\![\text{licked}]\!] = \lambda x. \lambda y. \lambda g. \begin{cases} \{g\} \text{ if } \textit{licked}'(x)(y) \\ \text{else } \emptyset \end{cases}$$

In both cases, what's returned is simply the unchanged input assignment if the standard truth-condition is met. Otherwise, nothing is returned, and we can be said to have observed a **failure**.

4.2 DPs

Proper names and pronouns will have a type reminiscent of quantificational DPs (or LIFTed non-quantificational DPs), i.e. $\langle\langle e, \pi\rangle, \pi\rangle$.

A proper name takes a dynamic predicate and feeds it an individual and an updated assignment function:

$$[\![\text{Bill}_n]\!] = \lambda P. \lambda g. P(\text{b})(g^{[\text{b}/n]})$$

Pronouns work similarly, but they feed a dynamic predicate an assignment-dependent individual and don't change the assignment function (exercise: what happens if '(g)' is replaced with '$(g[g(n)/n])$'?):

$$[\![\text{she}_n]\!] = \lambda P. \lambda g. P(g(n))(g)$$

Notice we need no special rule for interpreting pronouns. They just have a lexical semantics that mentions assignment functions, and that is that. Before, we had a special rule for interpreting pronouns (and traces).

Exercise: Why can't pronouns and proper names be type e? And is it really necessary to suppose that dref introduction is part of the *lexical semantics* of DPs?

4.3 Examples

We'll assume only **FA** for now (exercise: is **PM** definable when the propositional type is π?). Notice that this is our *old* notion of **FA**, i.e. the one that doesn't mention assignment functions.

Here's a case with a proper name. The result is equivalent to the meaning proposed in the previous section.

$[\![\text{Bill}_n \text{ left}]\!]$	$= [\![\text{Bill}_n]\!]([\![\text{left}]\!])$	**FA**
	$= \textbf{bill}_n(\textbf{left})$	Lexicon
	$= (\lambda P. \lambda g. P(\text{b})(g^{[\text{b}/n]}))(\textbf{left})$	Definition
	$= \lambda g. \textbf{left}(\text{b})(g^{[\text{b}/n]})$	β
	$= \lambda g. \{g^{[\text{b}/n]}\} \text{ if } \textit{left}'(\text{b}), \text{ else } \emptyset$	Definition, $\beta \times 2$

Here's a pronominal example. Again, result is equivalent to the meaning proposed in the previous section.

$[\![\text{he}_n \text{ left}]\!]$	$= [\![\text{he}_n]\!]([\![\text{left}]\!])$	**FA**
	$= \textbf{he}_n(\textbf{left})$	Lexicon
	$= (\lambda P. \lambda g. P(g(n))(g))(\textbf{left})$	Definition
	$= \lambda g. \textbf{left}(g(n))(g)$	β
	$= \lambda g. \{g\} \text{ if } \textit{left}'(g(n)), \text{ else } \emptyset$	Definition, $\beta \times 2$

5 Dynamic conjunction

Basic intuition: function composition. The left one returns an assignment, passes it to the right one.

$$r(l(g))$$

Just folding in sets: *relational* composition.

$$[\![\text{and}]\!] = \lambda r.\lambda l.\lambda g.\{h : k \in l(g) \text{ and } h \in r(k)\}$$

I find the iconic notation illuminating:

$$g[\text{S}_1 \text{ and } \text{S}_2]h \text{ iff } \exists k.g[\text{S}_1]k[\text{S}_2]h$$

Again order-sensitive, non-commutative. The updated assignments from the *left* conjunct are used to evaluate the *right* conjunct, and not vice versa.

Here's what this gives for *Bill$_2$ left; he$_2$ was tired*:

$$\lambda g.\{h : k \in [\![\text{Bill}_2 \text{ left}]\!](g) \text{ and } h \in [\![\text{he}_2 \text{ was tired}]\!](k)\}$$

Assuming Bill left, in which case $[\![\text{Bill}_2 \text{ left}]\!]$ simply returns $g^{[\text{b}/2]}$:

$$\lambda g.\{h : h \in [\![\text{he}_2 \text{ was tired}]\!](g^{[\text{b}/2]})\}$$

Assuming he was tired:

$$\lambda g.\{g^{[\text{b}/2]}\}$$

The discourse referent lives to fight another day! Notice that if either Bill left or wasn't tired, the evaluation gets short-circuited, and no assignments are returned.

6 Adding in indefinites

Remember the issue with indefinites: *which* discourse referent gets introduced by e.g. *a linguist*? As it happens, the relational perspective actually offers a ready solution problem! We *don't need to make a choice*!

That is, indefinites cause a *multitude* of updated assignment functions to be output.

$$[\![\text{a linguist}_n]\!] = \lambda P.\lambda g.\bigcup\{P(x)(g^{[x/n]}) : \mathit{ling}'(x)\}$$

This is a new bit of notation. It's an **infinitary union**. Don't be scared. All this formula says is to do what you did for Bill, but once for each linguist, and then collect the results. For example, if the linguists are Bob and Polly, then:

$$[\![\text{a linguist}_n]\!] = \lambda P.\lambda g.P(\text{b})(g^{[\text{b}/n]}) \cup P(\text{p})(g^{[\text{p}/n]})$$

Nothing about our basic setup needs to change to accommodate this possibility. We've already assumed that sentences output sets of assignments, so everything will type out. The only difference: indefinites allow there to be potentially multiple outputs!

7 Dynamically closed things

Recall data such as (2). Assume (for the sake of simplicity) that the structure of the first sentence there is (3).

(2) I don't own a car$_i$. *It$_i$'s a Hyundai.

(3) [not [$_\Sigma$ I own a car]]

As we've seen, Σ introduces a discourse referent. This suggests that the role of negation is to, in Karttunen terms, *delete* any discourse referents introduced by its complement Σ. Here's one way to do this:

$$g[\text{not S}]h \text{ iff } g = h \text{ and } g[\text{S}] = \emptyset$$

Less iconically:

$$[\![\text{not}]\!] = \lambda p.\lambda g.\begin{cases}\{g\} \text{ if } p(g) = \emptyset \\ \text{else } \emptyset\end{cases}$$

In other words, negation both requires that its complement yields a failure (that is, has no ouputs) and that the assignment function ultimately returned is simply the unchanged input assignment. There is no chance for any drefs that may be generated in the complement to make it out alive.

Exercise: what are the implications of this sort of theory of negation for cases like (4). How about cases like (5) and (6)?

(4) I don't like Bill$_i$. He$_i$'s boring.

(5) I don't like the man who criticized Barack$_i$. He$_i$'s a good president.

(6) It isn't the case that I don't own a radio$_i$. It$_i$'s a Panasonic.

A variety of other meanings are taken to be dynamically closed—that is, built on negation:

(7) If someone$_i$ knocked, she$_i$ left. *I saw her$_i$.

(8) Harvey courts a different woman$_i$ at every convention. *She$_i$'s a climatologist.