## Dynamics via scope

Dylan Bumford
April 15, 2015

### 1 Today

- First up, review the Montagovian account of sequence-based dynamic semantics
- Then we'll fiddle with this representation a little bit until it starts to look very close to the alternatives fragment that we've been working with
- With that intriguing correspondence in sight, we'll investigate how we might push the mechanisms for scope-taking explored in the last two weeks through the dynamic looking glass
- Once we've retraced our formal steps and established some scopal apparatus for dynamic constituents, we'll switch over to the empirical domain taken up in Bumford 2015: pair-lists
- Finally we'll see some applications of the dynamics+scope techniques that help make sense of the pair-list data

### 2 Super quick dynamic semantics review

- Following PLA, let sentences denote update functions: $\mathbf{T} \coloneqq \{\sigma\} \to \{\sigma\}$
- Relations are tests on inputs, of type $e \to \cdots \to \mathbf{T}$:

$$\mathbf{left} = \lambda x S.\ \begin{cases} S & \text{if } \mathsf{left}(x) \\ \varnothing & \text{otherwise} \end{cases}$$

$$\mathbf{met} = \lambda x y S.\ \begin{cases} S & \text{if } \mathsf{met}(x)(y) \\ \varnothing & \text{otherwise} \end{cases}$$

- Indefs and names introduce drefs, of type $(e \to \mathbf{T}) \to \mathbf{T}$:

$$\mathbf{Polly} = \lambda k S.\ k(\mathsf{p})(S \cdot \mathsf{p})$$

$$\mathbf{a.ling} = \lambda k S.\ \bigcup_{x \in \mathsf{ling}} k(x)(S \cdot x)$$

- Pronouns are just like names, except they find their referents in the context:

$$\mathbf{pro}_n = \lambda k S.\ \bigcup_{s \in S} k(s_n)(s)$$

- Finally, other GQs are externally static (dynamically closed). Like relations they just check to make sure that the input satisfies some property (their nuclear scope) when run at all/three/none/etc. of the members of their restrictors:

$$\mathbf{every.ling} = \lambda k S.\ \begin{cases} S & \text{if } \forall x \in \mathsf{ling}.\ k(x)(S) \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases}$$

### 3 Refactoring

- Right now, the denotations of our sentences are balanced, in the sense that they output the same type of thing that they receive as input: *sets of dref sequences*.
- This makes it very simple to *compose* updates, which is always a good thing.
- But to some extent it masks the way that updates are inherently *nondeterministic*; they have the power to introduce uncertainty even where none was before.
- And in any case, we're almost always going to begin our micro-discourses with a singleton set containing an empty sequence. Can we refactor things so that updates wear their nondeterminism somewhere closer to their sleeves (i.e. their types)? We can! In fact, it's really pretty trivial.
- So now the type of sentences will be $\mathbf{T} \coloneqq \sigma \to \{\sigma\}$, a function from a single input context to a potential multiplicity of output contexts. The rest is just pushing this through the system. Notice that a lot of things

don't even change!

$$\textbf{left} = \lambda xs.\ \begin{cases}\{s\} & \text{if } \mathsf{left}(x)\\ \varnothing & \text{otherwise}\end{cases}$$

$$\textbf{met} = \lambda xys.\ \begin{cases}\{s\} & \text{if } \mathsf{met}(x)(y)\\ \varnothing & \text{otherwise}\end{cases}$$

$$\textbf{Polly} = \lambda ks.\ k(\mathsf{p})(s{\cdot}\mathsf{p})$$

$$\textbf{a.ling} = \lambda ks.\ \bigcup_{x\in\mathsf{ling}} k(x)(s{\cdot}x)$$

$$\textbf{pro}_n = \lambda ks.\ k(s_n)(s)$$

$$\textbf{every.ling} = \lambda ks.\ \begin{cases}\{s\} & \text{if } \forall x \in \mathsf{ling}.\ k(x)(s) \neq \varnothing\\ \varnothing & \text{otherwise}\end{cases}$$

- What about conjunction? Have we lost that nice composability now that the inputs and outputs are unbalanced? Not really; we just have to compose *pointwise* (technically, this is "relation composition", rather than "function composition"):

$$\textbf{and} = \lambda mns.\ \bigcup_{s'\in m(s)} n(s')$$

## 4 Back to alternatives for a second

- Recall the suggestion from last week to treat sentences as functions *from assignments* to truth values, rather than plain truth values *parameterized* by assignments:

$$[\![\text{John met her}_3]\!]^g = \mathsf{T} \text{ iff } \mathsf{met}(\mathsf{j})(g(3))$$
$$\rightsquigarrow$$
$$[\![\text{John met her}_3]\!] = \lambda g.\ \mathsf{T} \text{ iff } \mathsf{met}(\mathsf{j})(g(3))$$

- Also recall that in the face of indeterminacy introduced by alternative generators, we'll often need to wrap sentences into singleton sets.

$$\left[\!\!\left[\boxed{\text{John met her}}_3\right]\!\!\right] = \lambda g.\ \{\mathsf{T}\} \text{ iff } \mathsf{met}(\mathsf{j})(\mathsf{m})$$

- Taking a cue from the dynamic fragment, we could build this boxing procedure into the meanings of verbs, in anticipation of any scoped indefinites. That would look something like this:

$$\textbf{left} = \lambda xg.\ \begin{cases}\{\mathsf{T}\} & \text{if } \mathsf{left}(x)\\ \{\mathsf{F}\} & \text{otherwise}\end{cases}$$

  Compare this to the dynamic entry in the previous column.

- In both fragments, a sentence denotes a function from some sort of environment to a set of results. But …
  - They seem to be fixated on different notions of an environment. The alternatives fragment is sensitive to contexts that manage variables (so that we can interpret traces). The dynamic fragment is sensitive to contexts that manage drefs (so that we can interpret pronouns).
  - They also seem to have different ideas about what a "result" is. The alternatives fragment returns a truth value (or more generally, some component of the model). The dynamic fragment returns a new context.

- In other words, as Simon has hinted before, the alternatives appear to be designed to make sense of indeterminacy in situations where scope-taking is at issue, while the dynamics appear to be designed to make sense of indeterminacy in situations where binding is at issue. But what if we need to do both?

- Here's a very simple idea. Let sentences read in a context and then return a set of *pairs*, such that those pairs contain a model-theoretic value **and** an updated context! Thus we redefine the return type of sentence-level expressions one last time: $\mathbf{T} := \sigma \to \{\langle t, \sigma\rangle\}$.

$$\textbf{left} = \lambda xs.\ \begin{cases}\{\langle \mathsf{T}, s\rangle\} & \text{if } \mathsf{left}(x)\\ \{\langle \mathsf{F}, s\rangle\} & \text{otherwise}\end{cases}$$

$$\textbf{met} = \lambda xys.\ \begin{cases}\{\langle \mathsf{T}, s\rangle\} & \text{if } \mathsf{met}(x)(y)\\ \{\langle \mathsf{F}, s\rangle\} & \text{otherwise}\end{cases}$$

- Or more concisely

$$\textbf{left} = \lambda xs\,.\,\{\langle \mathsf{left}(x), s\rangle\}$$
$$\textbf{met} = \lambda xys\,.\,\{\langle \mathsf{met}(x)(y), s\rangle\}$$

- Now we once more Notice again how the denotations of indefinites, names, and pronouns are exactly as before. It's worth thinking about why that should be. Only conjunction and the universal quantifier get a facelift.

$$\textbf{Polly} = \lambda ks\,.\,k(\mathsf{p})(s{\cdot}\mathsf{p})$$
$$\textbf{a.ling} = \lambda ks\,.\,\bigcup_{x\in\mathsf{ling}} k(x)(s{\cdot}x)$$
$$\textbf{pro}_n = \lambda ks\,.\,k(s_n)(s)$$
$$\textbf{every.ling} = \lambda ks\,.\,\{\langle \forall x \in \mathsf{ling}\,.\,k(x)(s) \neq \varnothing, s\rangle\}$$
$$\textbf{and} = \lambda mns\,.\,\{\langle p \wedge q, s''\rangle \mid \langle p, s'\rangle \in m(s),\ \langle q, s''\rangle \in n(s')\}$$

## 5 What about scope?

- We've already made such strides in finding a way to give indeterminate constituents (and constituents containing any uncaptured indeterminacy anywhere in them) *scope* without getting our wires all crossed when it comes to predicate abstraction.
- Recall from the alternatives fragment:

$$\boxed{x} = \{x\} \qquad\qquad \mathcal{A}^{\Uparrow} = \lambda k\,.\,\bigcup_{a\in\mathcal{A}} k(a)$$

- It's easy to see how we might generalize $\boxed{\cdot}$:

$$\boxed{x} = \lambda s\,.\,\{\langle x, s\rangle\}$$

- And working out the formula for $\Uparrow$ is not all that challenging either, if you take the previous $\Uparrow$ as a starting point and follow the types:

$$\mathcal{A}^{\Uparrow} = \lambda ks\,.\,\bigcup_{\langle a, s'\rangle\in\mathcal{A}} k(a)(s')$$

- But does this really work the same? Yes! Check it out:

$$\begin{aligned}
&\left(\lambda s\,.\,\{\langle \mathsf{die}(x), s{\cdot}x\rangle \mid \mathsf{rel}(x)\}\right)^{\Uparrow}\ \left(\lambda p\,.\,\boxed{p \Rightarrow \mathsf{house}}\right)\\
&= \left(\lambda s\,.\,\{\langle \mathsf{die}(x), s{\cdot}x\rangle \mid \mathsf{rel}(x)\}\right)^{\Uparrow}\ \left(\lambda ps\,.\,\{\langle p \Rightarrow \mathsf{house}, s\rangle\}\right)\\
&= \left(\lambda ks\,.\,\bigcup_{x\in\mathsf{rel}} k(\mathsf{die}(x))(s{\cdot}x)\right)\ \left(\lambda ps\,.\,\{\langle p \Rightarrow \mathsf{house}, s\rangle\}\right)\\
&= \lambda s\,.\,\{\langle \mathsf{die}(x) \Rightarrow \mathsf{house}, s{\cdot}x\rangle \mid \mathsf{rel}(x)\}
\end{aligned}$$

- Just as before, when we pied-pipe the antecedent above the conditional, the indeterminacy it subsumes bubbles out, so that we get a *set* of conditionals for our final denotation, one for each relative of mine. In addition, those conditionals are each *tagged* with an updated context that records which individual it depends on.

## 6 Universals and pair-lists

Distributive universal quantifiers that outscope alternative generators often give rise to "pair-list phenomena". This is easiest to characterize by example.

- Quantifying into questions

(1) Which language did every boy study?
   a. Japanese
   b. His mother tongue
   c. Al Arabic, Bill Basque, Carl Czech

(2) Which language did {these, most, several, no}boy(s) study?
   a. Japanese
   b. Their mother tongue
   c. # Al Arabic, Bill Basque, Carl Czech

Zooming in on 'every' vs. 'no':

(3) Which language did no boy remember to study?

a. # Al Arabic, Bill Basque, Carl Czech

(4) Which language did every boy forget to study?

a. Al Arabic, Bill Basque, Carl Czech

- Schlenker clauses, or "arbitrary functional" readings

(5) If each boy studied a certain language, then the exam was a sure success
$\exists f : \mathsf{boy} \to \mathsf{lang}\,.\,(\forall x \in \mathsf{boy}\,.\,\mathsf{study}(f(x))(x)) \Rightarrow \mathsf{success}$

(6) # If {these, most, several, no} boy(s) studied a certain language, then the exam was a sure success
$\exists f : \mathsf{boy} \to \mathsf{lang}\,.\,(\iota/\exists_\theta/\neg\exists x \in \mathsf{boy}\,.\,\mathsf{study}(f(x))(x)) \Rightarrow \mathsf{success}$

Zooming in on 'every' vs. 'no'

(7) If every reel lands on a certain symbol, you'll win the prize
$\exists f : \mathbb{R} \to S\,.\,(\forall x \in \mathbb{R}\,.\,\mathsf{land}(f(x))(x)) \Rightarrow \mathsf{win}$

(8) # If no reel lands on a certain symbol, you'll certainly lose
$\exists f : \mathbb{R} \to S\,.\,(\neg\exists x \in \mathbb{R}\,.\,\mathsf{land}(f(x))(x)) \Rightarrow \mathsf{lose}$

- Internal readings of comparative adjectives (these are fun, but we probably won't get to them)

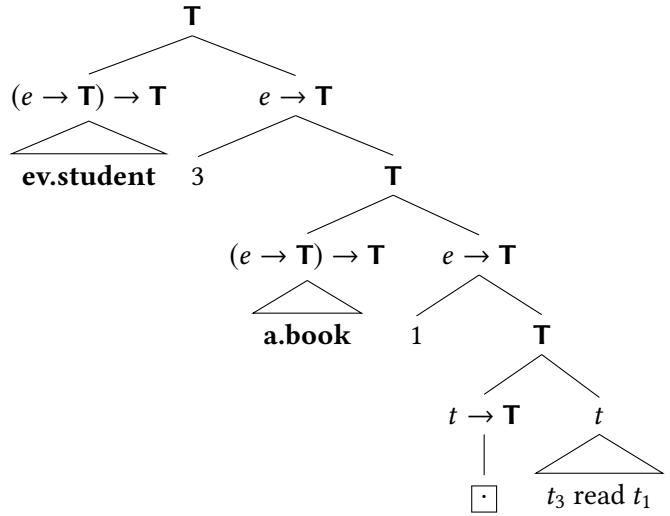
(9) Each guest brought a {different, more elaborate} dish

(10) # {These, Most, Several, No} guests brought a {different, more elaborate} dish

- The theory proposed in Bumford 2015: universals trigger iterated (dynamic) conjunction over the alternatives in their scope

$$\textbf{every.student} = \lambda k\,.\,{;}\{\lambda s'\,.\,k(x)(s'{\cdot}x) \mid x \in \mathsf{student}\}$$

(11) Every student read a book




$$\big(\lambda k\,.\,{;}\{\lambda s'\,.\,k(v)(s'{\cdot}v) \mid v \in \mathsf{stud}\}\big)\,\big(\lambda vs\,.\,\{\langle \mathsf{read}(y)(x), s{\cdot}v{\cdot}y\rangle \mid y \in \mathsf{book}\}\big)$$
$$= {;}\{\lambda s'\,.\,\{\langle \mathsf{read}(y)(v), s{\cdot}v{\cdot}y\rangle \mid y \in \mathsf{book}\} \mid v \in \mathsf{stud}\}$$
$$= \lambda s\,.\,\{\langle \mathsf{read}(x)(\mathsf{j}) \wedge \mathsf{read}(y)(\mathsf{m}) \wedge \mathsf{read}(z)(\mathsf{f}), s{\cdot}\mathsf{j}{\cdot}x{\cdot}\mathsf{m}{\cdot}y{\cdot}\mathsf{f}{\cdot}z\rangle \mid x, y, z \in \mathsf{book}\}$$

## 7 Putting the theory to work

- Notice that iterated conjunction multiplies alternatives rather than squashing them! Compare:

$$\lambda s\,.\,\{\langle \forall x \in \mathsf{student}\,.\,\exists y \in \mathsf{book}\,.\,\mathsf{read}(y)(x), s\rangle\}$$

with

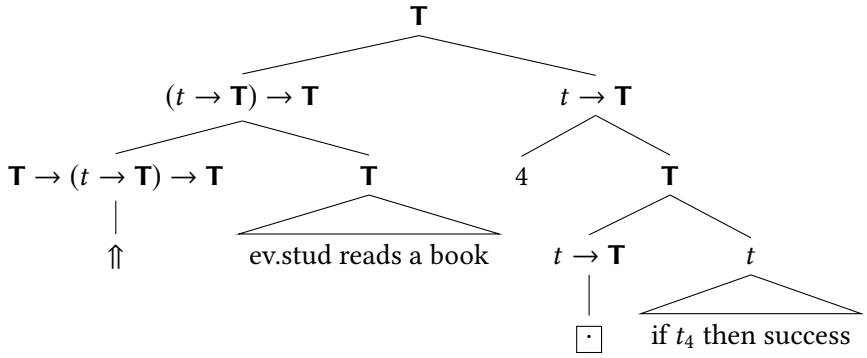
$$\lambda s\,.\,\{\langle \mathsf{read}(x)(\mathsf{j}) \wedge \mathsf{read}(y)(\mathsf{m}) \wedge \mathsf{read}(z)(\mathsf{f}), s{\cdot}\mathsf{j}{\cdot}x{\cdot}\mathsf{m}{\cdot}y{\cdot}\mathsf{f}{\cdot}z\rangle \mid x, y, z \in \mathsf{book}\}$$

- This has two interesting consequences from the perspective of this course:
  - It means quantifying *universally* into questions preserves their "inquisitiveness" (pair-list questions are still questions), in contrast with other GQs, which are dynamically closed, and as a result, non-inquisitive.

  - It means that when we pied-pipe whole sentences above some operator, the indeterminacy generated by the iterated conjunction shoud *outscope* the things it is raised over.

- Let me illustrate with the latter

(12) If every student reads a certain book, the exam will be a success




$[\![(11)]\!]^{\Uparrow}\ (\lambda ps\,.\,\{\langle p \Rightarrow \text{success}, s\rangle\})$

$= \left(\lambda ks\,.\,\bigcup_{\langle p, s'\rangle \in [\![(11)]\!]} k(p)(s')\right)(\lambda ps\,.\,\{\langle p \Rightarrow \text{success}, s\rangle\})$

$= \lambda s\,.\,\{\langle p \Rightarrow \text{success}, s'\rangle \mid \langle p, s'\rangle \in [\![(11)]\!]\}$

$\approx$ for some way of pairing up students with books, if every student the book he's paired with, the exam will be a success

## 8 Concluding

- The techniques developed for handling alternatives in a scope-flexible grammar extend elegantly and immediately to dynamic representations

- Our LF pied-piping story turns out to be exactly what the doctor ordered for exceptionally scoping pair-lists, which arise from an interaction of multiple semantic componenents within an island (and thus can't be reduced to a fancied up denotation for some particular lexical item). In other words, how could we have possibly done this without ⇑ without overgenerating?

- I've floated the idea that distributive universals are really conjunctive iterators. There are a few arguments for this that don't have to do with scope, but it'd be good to know what else this predicts!

- Note that despite the symmetries between the alternatives fragment with abstracted assignments and dynamic fragments like PLA, I've quietely retreated to using assignments as parameters over constituents, rather than genuine arguments to denotations. It is not easy to get around this in a framework with traces, but also not impossible …