LING419 Homework 2
Interpretation of trees and movement

For this assignment, when asked to give a derivation in the style of Heim and Kratzer (H&K), you should give a semantic value and a semantic type for each node of the tree, like this:

\[
\text{saw}(\text{mary}')(\text{john}')
\]

\[
\lambda y.\text{saw}(\text{mary}')(y) \quad e \rightarrow t
\]

\[
\lambda x\lambda y.\text{saw}(x)(y) \quad \text{mary}'
\]

\[
e \rightarrow (e \rightarrow t)
\]

There are two rules in the H&K system (for now). Wherever two constituents combine to form a larger one, you need to use one of these rules to determine the semantic value of the new, larger constituent. In the simple example above, function application applies at each node. Remember that trees are like mobiles, so left-to-right order is insignificant. In the following descriptions, \(\alpha\) and \(\beta\) are placeholders for semantic types like \(e\), \(t\), \(e \rightarrow t\), etc.

- **Function application** (FA): If you combine something denoting \(f\), which is a function from type \(\alpha\) to type \(\beta\), with something denoting \(x\), which is of type \(\alpha\), the result denotes \(f(x)\) — which is, of course, of type \(\beta\).

  Recipe:

  \[
  f(x) \\
  \beta
  \]

  Example:

  \[
  \lambda y.\text{saw}(\text{mary}')(y) \\
  e \rightarrow t
  \]

  \[
  \lambda x\lambda y.\text{saw}(x)(y) \quad \text{mary}'
  \]

  \[
  e \rightarrow (e \rightarrow t)
  \]

- **Predicate abstraction** (PA): If you combine something denoting \(\Phi\) with an index \(i\), the result denotes what you get when you: (a) choose a new variable which doesn’t appear anywhere in \(\Phi\), say \(v\); (b) put \(\lambda v\) on the front of \(\Phi\); and (c) wherever \(t_i\) appears in \(\Phi\), replace it with \(v\).

  Recipe:

  \[
  \lambda v...v... \\
  i...t_i...t_i...
  \]

  Example:

  \[
  \lambda x.\text{slep}t'(x) \\
  1
  \]

Finally: in this system, we take the semantic value of a trace \(t_i\), for any index \(i\), to be a sort of “dummy entity” of type \(e\), written just as \(t_i\).\(^1\)

---

\(^1\)In the Heim and Kratzer book they write \(A(i)\) as the semantic value of a trace \(t_i\), and if you did semantics last semester you probably used this too. In class, I sometimes wrote \(\Box_i\). If you want to use one of those here, that’s fine. It makes no difference for our purposes.
A  Basic function application (3 points)

Give the full H&K-style derivation for the sentence in (1), using the tree in (2) and the lexical semantic values in (3).

(1) Mary met Bill if John slept

(2)

Mary
met
Bill
if
John
slept

(3)

[Mary] := mary′
[met] := λxλy.met′(x)(y)
[if] := λxλy.x ⇒ y
[Bill] := bill′
[slept] := λx.slept′(x)
[John] := john′

You’ll only need the FA rule to do this. If you have trouble getting started, try to write down the semantic type of each of the lexical values in (3). The type of [if] is t → (t → t).²

B  Relative clauses (7 points)

Relative clauses in English

Give the full H&K-style derivation for the expression in (4), using the tree in (5) and the lexical semantic values in (6).

Notice that this expression is just a phrase — not a complete sentence — so it will not end up with a semantic value of type t. Rather, since the distribution of this phrase is just the same as the distribution of the word ‘girl’, the final semantic value of the phrase should be of type e → t, like ‘girl’ is.

(4) girl who John met

(5)
girl
who
John
met

(6)

[John] := john′
[girl] := λx.girl′(x)
[who] := λPλQλx.P(x) ∧ Q(x)

For comparison, fill in the semantics of the following CCG derivation for the same expression:

²Don’t confuse the “single arrow” that we’re using to write function types (eg. e → t for the type of functions from e to t) with the “double arrow” used in the lexical value for ‘if’. This double arrow is a symbol of logic, like ∧ and ∨. The details aren’t actually important for this question, but you can think of it as: just as “P and Q” is written in logic as P ∧ Q and “P or Q” is written as P ∨ Q, “P implies Q” is written as P ⇒ Q. The single arrow is usually used to mean both things, but in this assignment at least I’ll stick to using the single arrow for function types, and the double arrow for logical implication. (It isn’t an accident that the same symbol gets used for both of these two things — deep down they’re the same thing in two different guises — but it still has the potential to confuse at first, so I’ll avoid it.)
Relative clauses in Whinglish

We’ve seen that there’s good reason to believe that the behaviour of words like ‘who’ and ‘what’ in wh-in-situ languages, like Chinese and Whinglish, is actually much more similar to their behaviour in languages like English than it first appears.

Give the full H&K-style derivation for the semantics of the Whinglish phrase in (7) (equivalent to the English phrase in (4)). Assume that the semantic values of the Whinglish words are identical to those of the English equivalents given in (6). You will need to draw a tree structure that does not correspond to the order in which the words are pronounced in Whinglish.

(7)  GIRL JOHN MET WHO

Hint: This question is much easier than it probably seems at first.

C  Quantification (5 points)

It turns out that the exact same mechanisms as we found ourselves forced to adopt for Whinglish relative clauses — given findings about in-situ wh-words being not completely in-situ after all — can be used for quantification in English.

Give the full H&K-style derivation for the sentence in (8), using the tree in (9) and the lexical semantic values in (10). We’ll pretend that ‘exactly three’ is just a single word.

Recall that if $P$ is some property, then $\{x : P(x)\}$ is the set of all things that have that property. For example, $\{x : \text{girl}(x)\}$ is the set of all girls.3

(8)  John met exactly three linguists

(9)

3See Partee et al. chapter 1, if you’re unsure about this. They use the notation $\{x \mid P(x)\}$ to mean exactly the same thing. I prefer the colon version because it avoids any possibility of confusion with cardinality bars.
(10)  
\[[\text{John}] := \text{\textit{john}}\] \quad  
\[[\text{met}] := \lambda x \lambda y. \text{\textit{met}}(x)(y)\] 
\[[\text{linguists}] := \lambda x. \text{\textit{linguist}}(x)\] \quad  
\[[\text{exactly three}] := \lambda P \lambda Q. |\{x : P(x)\} \cap \{x : Q(x)\}| = 3\] 

For comparison, fill in the semantics of the following CCG derivation for the same sentence:

\[
\begin{array}{c|c|c|c}
\text{John} & \text{met} & \text{exactly three} & \text{linguists} \\
\text{NP} & \text{NP} & \text{NP} & \text{NP} \\
\text{john} & \lambda x \lambda y. \text{\textit{met}}(x)(y) & \lambda P \lambda Q. |\{x : P(x)\} \cap \{x : Q(x)\}| = 3 & \lambda x. \text{\textit{linguist}}(x) \\
\text{S/(S/NP)} & \text{S/(S/NP)} & \text{S/(S/NP)} & \text{S/(S/NP)} \\
\ldots & \ldots & \ldots & \ldots \\
\end{array}
\]

D  Quantifier scope ambiguities (5 points)

Give two distinct H&K-style derivations for the string of words in (11), using the lexical semantic values in (13). One derivation should end up pairing it with the meaning in (12a), and the other should end up pairing it with the meaning in (12b).

(11)  Some linguist met every boy
(12)  a.  \(\exists x [\text{\textit{linguist}}(x) \land \forall y [\text{\textit{boy}}(y) \Rightarrow \text{\textit{met}}(y)(x)]]\)  
    b.  \(\forall y [\text{\textit{boy}}(y) \Rightarrow \exists x [\text{\textit{linguist}}(x) \land \text{\textit{met}}(y)(x)]]\)

\[[\text{linguist}] := \lambda x. \text{\textit{linguist}}(x)\] \quad  
\[[\text{some}] := \lambda P \lambda Q. \exists x [P(x) \land Q(x)]\] 
\[[\text{boy}] := \lambda x. \text{\textit{boy}}(x)\] \quad  
\[[\text{every}] := \lambda P \lambda Q. \forall x [P(x) \Rightarrow Q(x)]\] 
\[[\text{met}] := \lambda x \lambda y. \text{\textit{met}}(x)(y)\]

One of the derivations will use the tree in (14); the other will use a different tree.

(14)

Extra: (Not for extra course credit, just for the fun and satisfaction.) We’ve written the semantic value of ‘every’ and ‘some’ using logical symbols like \(\forall\) and \(\exists\) and so on, without talking about sets; when talking about conservativity, and for ‘exactly three’ above, we used sets. We can also write the semantics for ‘every’ and ‘some’ using sets. Can we write the semantics for ‘exactly three’ (or ‘exactly one’ or ‘exactly two’) using logical symbols like \(\forall\) and \(\exists\) and so on? How about ‘most’?