Homework 4: Solutions

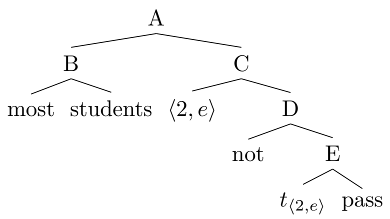
1 Determiner meanings

For any property P, I will use P_S to denote the set characterized by P, i.e. $\{x : P(x)\}$. For those of you who know some first-order logic, I'll include equivalent first-order formalizations where possible (those of you who don't know first-order logic so well may still find it instructive to consider these alternative renderings).

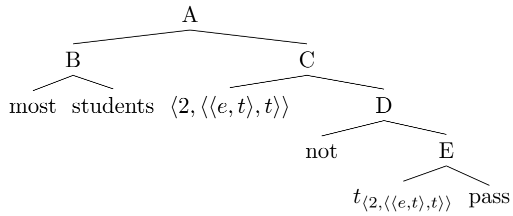
- $[\![\text{every}]\!]^g = \lambda P.\lambda Q.P_S \subseteq Q_S$
  Alternative first-order formalization: $\lambda P.\lambda Q.\forall x.P(x) \Rightarrow Q(x)$
- $[\![\text{no}]\!]^g = \lambda P.\lambda Q.P_S \cap Q_S = \emptyset$
  Alternative first-order formalization: $\lambda P.\lambda Q.\neg\exists x.P(x) \wedge Q(x)$
- $[\![\text{three}]\!]^g = \lambda P.\lambda Q.|P_S \cap Q_S| \geqslant 3$
  Alternative first-order formalization: $\lambda P.\lambda Q.\exists_{\geqslant 3}x.P(x) \wedge Q(x)$
- $[\![\text{most}]\!]^g = \lambda P.\lambda Q.|P_S \cap Q_S| > \frac{|P_S|}{2}$
  Not expressible with first-order quantification.
- $[\![\text{at least 5 but no more than 10}]\!]^g = \lambda P.\lambda Q.5 \leqslant |P_S \cap Q_S| \leqslant 10$
  Alternative first-order formalization: $\lambda P.\lambda Q.(\exists_{\geqslant 5}x.P(x) \wedge Q(x)) \wedge (\exists_{\leqslant 10}x.P(x) \wedge Q(x))$

2 Reconstruction, covert subjects

- *most students didn't pass*
  - ▷ Say 50% of the students passed and 50% failed. Then the first reading (most students are such that they didn't pass) is false, and the second (it's false that most students passed) is true.
  - ▷ The two trees and derivations follow. The first uses a type-e trace and thereby derives an interpretation on which $[\![\text{most students}]\!]^g$ receives as its argument the property characterizing individuals who didn't pass. This derives the first reading. The second uses a higher-order trace, which causes *most students* to reconstruct into the scope of negation. This derives the second reading.

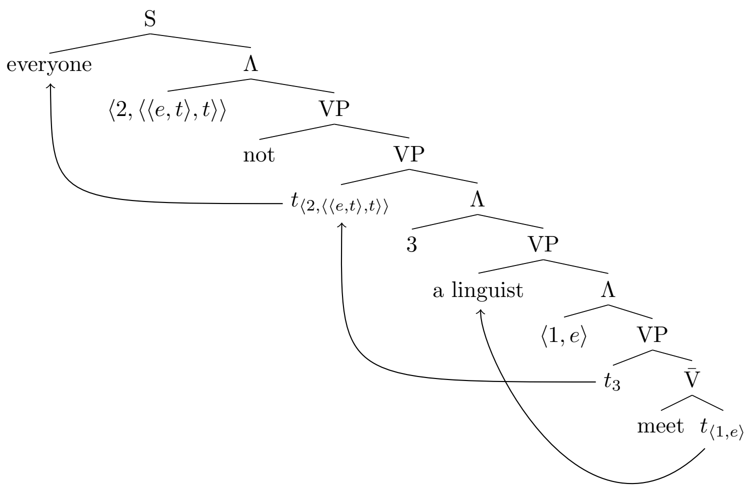


$$
\begin{aligned}
[\![\text{A}]\!]^g &= [\![\text{B}]\!]^g([\![\text{C}]\!]^g) && \textbf{FA}\\
&= [\![\text{B}]\!]^g(\lambda x.[\![\text{D}]\!]^{g[x/\langle 2,e\rangle]}) && \textbf{PA}\\
&= [\![\text{B}]\!]^g(\lambda x.[\![\text{not}]\!]^{g[x/\langle 2,e\rangle]}([\![\text{E}]\!]^{g[x/\langle 2,e\rangle]})) && \textbf{FA}\\
&= [\![\text{B}]\!]^g(\lambda x.\neg([\![\text{E}]\!]^{g[x/\langle 2,e\rangle]})) && \text{Lexicon}\\
&= [\![\text{B}]\!]^g(\lambda x.\neg([\![\text{pass}]\!]^{g[x/\langle 2,e\rangle]}([\![t_2]\!]^{g[x/\langle 2,e\rangle]}))) && \textbf{FA}\\
&= [\![\text{B}]\!]^g(\lambda x.\neg(\mathit{pass}'(x))) && \text{Lexicon, trace rule}\\
&= [\![\text{most}]\!]^g([\![\text{students}]\!])^g(\lambda x.\neg(\mathit{pass}'(x))) && \textbf{FA}\\
&= (\lambda P.\lambda Q.|P_S \cap Q_S| > \tfrac{|P_S|}{2})(\mathit{student}')(\lambda x.\neg(\mathit{pass}'(x))) && \text{Lexicon} \times 2\\
&= (\lambda Q.|\mathit{student}'_S \cap Q_S| > \tfrac{|\mathit{student}'_S|}{2})(\lambda x.\neg(\mathit{pass}'(x))) && \beta\\
&= |\mathit{student}'_S \cap (\lambda x.\neg(\mathit{pass}'(x)))_S| > \tfrac{|\mathit{student}'_S|}{2} && \beta\\
&= |\mathit{student}'_S \cap \{x : \neg(\mathit{pass}'(x))\}| > \tfrac{|\mathit{student}'_S|}{2} && \equiv
\end{aligned}
$$



$$
\begin{aligned}
[\![\mathrm{A}]\!]^g &= ([\![\mathrm{C}]\!]^g)([\![\mathrm{B}]\!]^g) && \textsf{FA}\\
&= (\lambda\nu.[\![\mathrm{D}]\!]^{g[\nu/\langle 2,\langle\langle e,t\rangle,t\rangle\rangle]})([\![\mathrm{B}]\!]^g) && \textsf{PA}\\
&= (\lambda\nu.[\![\mathrm{not}]\!]^{g[\nu/\langle 2,\langle\langle e,t\rangle,t\rangle\rangle]}([\![\mathrm{E}]\!]^{g[\nu/\langle 2,\langle\langle e,t\rangle,t\rangle\rangle]}))([\![\mathrm{B}]\!]^g) && \textsf{FA}\\
&= (\lambda\nu.\neg([\![\mathrm{E}]\!]^{g[\nu/\langle 2,\langle\langle e,t\rangle,t\rangle\rangle]}))([\![\mathrm{B}]\!]^g) && \text{Lexicon}\\
&= (\lambda\nu.\neg([\![t_2]\!]^{g[\nu/\langle 2,\langle\langle e,t\rangle,t\rangle\rangle]}([\![\mathrm{pass}]\!]^{g[\nu/\langle 2,\langle\langle e,t\rangle,t\rangle\rangle]})))([\![\mathrm{B}]\!]^g) && \textsf{FA}\\
&= (\lambda\nu.\neg(\nu(\mathit{pass}')))([\![\mathrm{B}]\!]^g) && \text{Lexicon, trace rule}\\
&= \neg([\![\mathrm{B}]\!]^g(\mathit{pass}')) && \beta\\
&= \neg([\![\mathrm{most}]\!]^g([\![\mathrm{students}]\!]^g)(\mathit{pass}')) && \textsf{FA}\\
&= \neg((\lambda P.\lambda Q.|P_S \cap Q_S| > \tfrac{|P_S|}{2})(\mathit{student}')(\mathit{pass}')) && \text{Lexicon} \times 2\\
&= \neg((\lambda Q.|\mathit{student}'_S \cap Q_S| > \tfrac{|\mathit{student}'_S|}{2})(\mathit{pass}')) && \beta\\
&= \neg(|\mathit{student}'_S \cap \mathit{pass}'_S| > \tfrac{|\mathit{student}'_S|}{2}) && \beta\\
&= |\mathit{student}'_S \cap \mathit{pass}'_S| \leqslant \tfrac{|\mathit{student}'_S|}{2} && \equiv
\end{aligned}
$$

- *everyone didn't meet a linguist*
  - ▷ Interpretation derived: *not* > *a linguist* > *everyone*—i.e. it's false that there's a linguist who everyone met.[1] Reason: the higher-order trace causes *everyone* to reconstruct to its pre-movement position. Meanwhile, the type-e trace means that *a linguist* is interpreted in its higher position, from which it c-commands the higher-order trace left by *everyone*. Thus, *a linguist* has scope over *everyone*, and both have scope under negation.
  - ▷ The following tree will do the job. We QR the trace of the overtly moved *everyone* to a position above *a linguist*. Moreover, when we do this, we leave a type-e trace. Thus, *everyone* reconstructs to within the scope of negation but still scopes over *a linguist*. Again, both quantifiers scope under negation.[2]

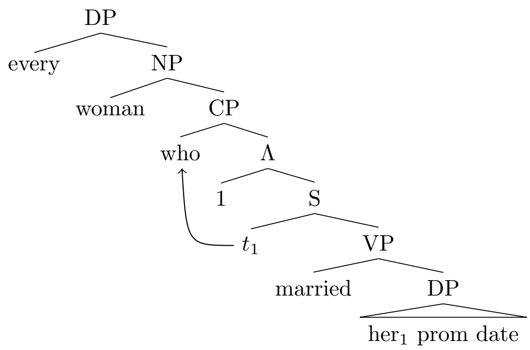


Notice, by the way, that there's no need to assign the same number to the higher-order and lower-order traces associated with *everyone*. The higher-order trace is bound by the higher-order abstractor, and the type-e trace is bound **separately** by a type-e abstractor.

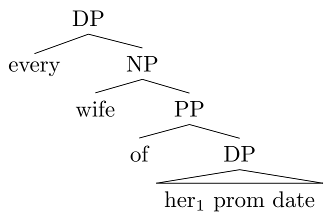
[1] I'm genuinely uncertain if this is a possible interpretation of this sentence.

[2] NB: there's no good way to draw the arrows in this tree. The way I've done it suggests that *everyone* moved out of the QR'd trace's LF position, but of course *everyone* moved out of the trace's surface position.

- **BONUS**
  - ▷ Here is how we derive the bound interpretation for the relative clause example (abstracting away from VP-internal subjects). Notice that the covert subject is absolutely crucial. For binding to happen, the covert subject and VP-internal pronoun *her* need to be co-indexed with each other and with the abstractor. This guarantees that property that results after abstraction characterizes individuals who married their prom date.

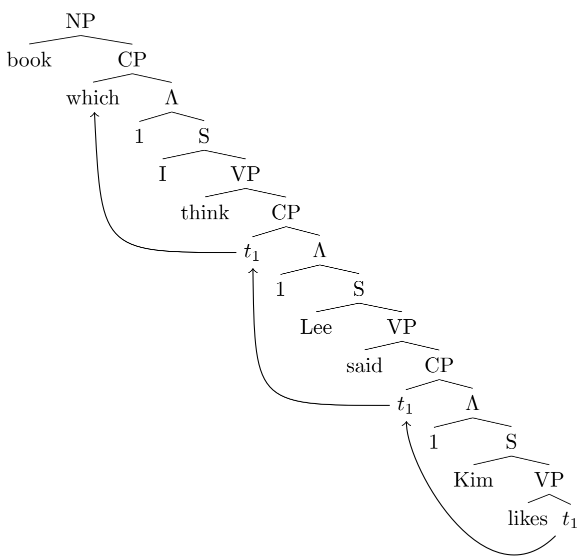


  - ▷ The ungrammaticality of the corresponding relational-noun case (even though the missing interpretation is a sensible one, as the relative-clause example makes clear) gives evidence against the covert-subject hypothesis—or, at least, absent an independent explanation, it gives evidence that relational nouns do not receive covert subjects. The lack of a covert subject prevents *her* from being bound within the DP (make sure you understand why simply plopping an abstraction index above NP won't work here):



3 Successive cyclic movement

- The problem: type clashes at both intermediate CPs. In each case, the S is type t, and the sister of S is type e. We have no way to compose these up and even if we did, there would be no way to assign these cases a reasonable semantics (short of ignoring the contribution of the trace).
- The solution: simply associate every intermediate landing site with an abstraction index! This lets the intermediate traces compose with their sisters seamlessly via functional application:



- Notice that there is no need to keep reapeating the index 1 down the tree. Each trace is only bound by the nearest c-commanding abstractor (recall the previous homework problem where we worked out why this was so), and so the following tree gives the same result as above: