ANLP Tutorial Exercise Set 5 (for tutorial groups in week 10)

v1.1
School of Informatics, University of Edinburgh
Sharon Goldwater

Exercise 1: First-order logic

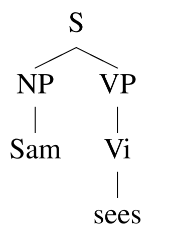
a) (a) False
   (b) $\{Zoot, Spot\}$
   (c) $\{(Mary, Zoot), (Li, Spot)\}$
   (d) $\{Spot\}$ (note this is a set, not just an entity)
   (e) True (because the antecedent is false)
   (f) False
   (g) False
   (h) True

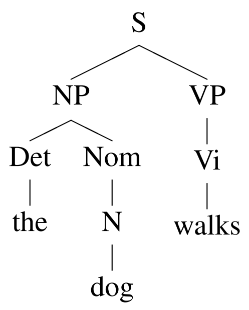
b) (a) All rabbits are furry
   (b) Franz helps Marie
   (c) Liang eats a sandwich
   (d) Liang eats a sandwich with a fork
   (e) All the students (separately) lift Marie (or "Each student lifts Marie"): there is a separate lifting event for each student
   (f) All the students (together) lift Marie: there is a single lifting event for all students

c) (a) $\exists e.hating(e) \wedge hater(e, Fiona) \wedge hatee(e, Ewan)$
   (b) There are two meanings, though not due to any form of ambiguity we've discussed before. The ambiguity here is caused by the verb. (a) Partha is currently eating a pizza: $\exists e.eating(e) \wedge eater(e, Partha) \wedge \exists x.pizza(x) \wedge eaten(e, x)$ (or perhaps we might want to think of "pizza" as a mass noun and treat it as an entity, yielding $\exists e.eating(e) \wedge eater(e, Partha) \wedge eaten(e, pizza)$). OR (b) Partha habitually or in principle eats pizza: It is much less clear how to translate this sentence into FOL, since our usual event semantics assume the existence of a single event with particular properties. In this case there are potentially many (or no) actual pizza-eating events. (I don't know how a semanticist would deal with this case, but to me it seems to share some of the same problems as so-called generic statements like "birds can fly" or "cats are furry", whose semantics are very much a subject of research.)
   (c) $\forall x.student(x) \Rightarrow \exists e.liking(e) \wedge liker(e, x) \wedge likee(e, Juan)$ (Note that placing the $\exists e$ outside the $\forall x$ would be a bit weird: it suggests that all students participate in a single collective liking event.
   (d) Quantifier scope ambiguity leads to two meanings. (a) There is a single student who likes all classes: $\exists x.student(x) \wedge \forall y.class(y) \Rightarrow \exists e.liking(e) \wedge liker(e, x) \wedge likee(e, y)$ OR (b) Each class is liked by at least one student: $\forall x.class(x) \Rightarrow \exists y.student(y) \wedge \exists e.liking(e) \wedge liker(e, y) \wedge likee(e, x)$
   (e) $\exists e.seeing(e) \wedge seer(e, Ella) \wedge seen(e, Ella)$
   (f) $\forall x.Tuesday(x) \Rightarrow \exists e.dancing(e) \wedge dancer(e, Ella) \wedge time(e, x)$ (or perhaps replace the final conjunct with something like $during(e, x)$).

Exercise 2: Semantic analysis

a) The syntactic trees are:





In the first tree, the meanings attached to each tree node are as follows:

- Sam, NP: $\lambda P.P(Sam)$
- sees, Vi, VP: $\lambda x.\exists e.seeing(e) \wedge seer(e,x)$
- S: derived as NP.sem(VP.sem), which is

$$\begin{aligned}(\lambda P.P(Sam))[\lambda x.\exists e.seeing(e) \wedge seer(e,x)] &= (\lambda x.\exists e.seeing(e) \wedge seer(e,x))[Sam] \\ &= \exists e.seeing(e) \wedge seer(e,Sam)\end{aligned}$$

I've added a little more notation to keep things clear: I put round brackets around the lambda expression that's about to undergo lambda reduction, and square brackets around the expression that's about to be substituted in place of the lambda variable. This will help especially with more complicated reductions below.

In the second tree, the meanings attached to each tree node are as follows:

- the, Det: $\lambda P.\lambda Q.\exists !x.P(x) \wedge Q(x)$
- dog, N, Nom: $\lambda x.dog(x)$
- walks, Vi, VP: $\lambda x.\exists e.walking(e) \wedge walker(e,x)$
- NP (the dog): derived as Det.sem(Nom.sem), which is

$$\begin{aligned}(\lambda P.\lambda Q.\exists !x.P(x) \wedge Q(x))[\lambda x.dog(x)] &= (\lambda P.\lambda Q.\exists !x.P(x) \wedge Q(x))[\lambda y.dog(y)] \quad (1)\\ &= \lambda Q.\exists !x.(\lambda y.dog(y))[x] \wedge Q(x) \quad (2)\\ &= \lambda Q.\exists !x.dog(x) \wedge Q(x) \quad (3)\end{aligned}$$

I did something important in line 1: before I substituted $\lambda x.dog(x)$ in for P, I re-named the variable x in the substituted expression and called it y. That's because there was already an x in the outer expression. In this case, those x's turn out to refer to the same thing, but that isn't guaranteed to be the case. If I didn't re-name the variable, I might have ended up making two variables that are supposed to refer to different things actually refer to the same thing. (This is basically a namespace issue, just like what happens when you define a variable inside a function in Python with the same name as one outside the function: implicitly, Python treats them as two different things. Here we have to rename one of them explicitly to avoid clashes.)

- S: derived as NP.sem(VP.sem), which is

$$\begin{aligned}&(\lambda Q.\exists !x.dog(x) \wedge Q(x))[\lambda x.\exists e.walking(e) \wedge walker(e,x)] \quad (4)\\ &= \exists !x.dog(x) \wedge (\lambda y.\exists e.walking(e) \wedge walker(e,y))[x] \quad (5)\\ &= \exists !x.dog(x) \wedge \exists e.walking(e) \wedge walker(e,x) \quad (6)\end{aligned}$$

Again, notice that I re-named the x in the "walk" part in the second line.

b) The problem is that the semantic attachment for VP says that the Vt takes the NP as its argument. But all our NPs have meanings like $\lambda P.(\ldots)$. If we apply the proposed simple (base-form) MR for `see` to something like that, we substitute the complex NP MR for y and end up with an invalid FOL expression. The solution is to *type-raise* the MRs for transitive verbs, just as we saw type-raising for NPs in lecture. This allows the Vt to take its NP argument and apply that NP to the base meaning of the Vt. For example, the MR for `walks Spot` is derived as follows.

$$\begin{aligned}
&(\lambda P \lambda x.P(\lambda y.\exists e.walking(e) \wedge walker(e,x) \wedge walkee(e,y)))[\lambda P.P(Spot)] && (7)\\
&= (\lambda P \lambda x.P(\lambda y.\exists e.walking(e) \wedge walker(e,x) \wedge walkee(e,y))[\lambda Q.Q(Spot)]) && (8)\\
&= \lambda x.(\lambda Q.Q(Spot))[\lambda y.\exists e.walking(e) \wedge walker(e,x) \wedge walkee(e,y)] && (9)\\
&= \lambda x.(\lambda y.\exists e.walking(e) \wedge walker(e,x) \wedge walkee(e,y))[Spot] && (10)\\
&= \lambda x.\exists e.walking(e) \wedge walker(e,x) \wedge walkee(e,Spot) && (11)
\end{aligned}$$

c) We just showed how to compute the meaning of the VP in "Sam walks Spot". We can combine it with the subject NP using the NP.sem(VP.sem) attachment as follows:

$$\begin{aligned}
&(\lambda P.P(Sam))[\lambda x.\exists e.walking(e) \wedge walker(e,x) \wedge walkee(e,Spot)] && (12)\\
&= (\lambda x.\exists e.walking(e) \wedge walker(e,x) \wedge walkee(e,Spot))[Sam] && (13)\\
&= \exists e.walking(e) \wedge walker(e,Sam) \wedge walkee(e,Spot) && (14)
\end{aligned}$$

To compute the MR for "Sam walks" we have:

$$\begin{aligned}
&(\lambda P.P(Sam))[\lambda x.\exists e.walking(e) \wedge walker(e,x)] && (15)\\
&= (\lambda x.\exists e.walking(e) \wedge walker(e,x))[Sam] && (16)\\
&= \exists e.walking(e) \wedge walker(e,Sam) && (17)
\end{aligned}$$

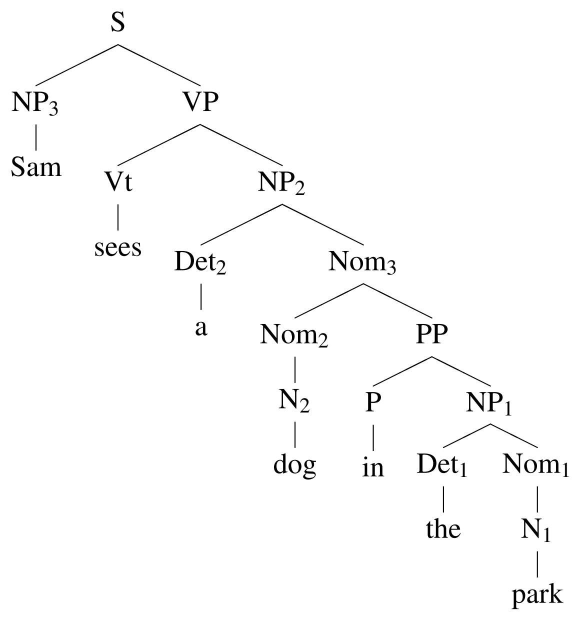
So, according to the meaning representations, "Sam walks" is entailed by "Sam walks Spot": in every case where the latter is true, the former will also be true. However, in common usage, this entailment does not hold. For example, I might walk a dog by standing with a leash and letting the dog run around, or by riding a bicycle with the dog on a leash. This is different from the pair "Sam sees Spot"/"Sam sees" because in this case, the second sentence really is entailed by the first one.

The issue with "walk" is that the transitive form actually has a slightly different meaning from the intransitive form. It means something like "make someone/something walk": the *object* of "walk" is the walker, not the subject. So we might consider changing the MR to something like:

$$\lambda P \lambda x.P(\lambda y.\exists e.walking(e) \wedge walker(e,y) \wedge instigator(e,x)) \qquad (18)$$

where y (which will be the object of the sentence) is now the walker instead of the walkee.

d) First, the analysis where PP attaches inside NP. I use subscripts so I can refer to the nodes when there are multiple ones with the same phrasal category.



The MRs derived at each node are as follows:

- park, N_1, Nom_1: $\lambda x.park(x)$
- the, Det_1: $\lambda P.\lambda Q.\exists! x.P(x) \wedge Q(x)$
- NP_1, as Det_1.sem(Nom_1.sem):

$$\begin{aligned}
(\lambda P.\lambda Q.\exists! x.P(x) \wedge Q(x))[\lambda y.park(y)] &= \lambda Q.\exists! x.(\lambda y.park(y))[x] \wedge Q(x) && (19)\\
&= \lambda Q.\exists! x.park(x) \wedge Q(x) && (20)
\end{aligned}$$

- in, P: $\lambda P.\lambda Q \lambda x.P(\lambda y.in(x,y)) \wedge Q(x)$
- PP (in the park), as P.sem(NP_1.sem):

$$\begin{aligned}
&(\lambda P.\lambda Q \lambda x.P(\lambda y.in(x,y)) \wedge Q(x))[\lambda Q.\exists! x.park(x) \wedge Q(x)] && (21)\\
&= (\lambda P.\lambda Q \lambda x.P(\lambda y.in(x,y)) \wedge Q(x))[\lambda R.\exists! z.park(z) \wedge R(z)] && (22)\\
&= \lambda Q \lambda x.(\lambda R.\exists! z.park(z) \wedge R(z))[\lambda y.in(x,y)] \wedge Q(x) && (23)\\
&= \lambda Q \lambda x.\exists! z.park(z) \wedge (\lambda y.in(x,y))[z] \wedge Q(x) && (24)\\
&= \lambda Q \lambda x.\exists! z.park(z) \wedge in(x,z) \wedge Q(x) && (25)
\end{aligned}$$

- dog, N_2, Nom_2: $\lambda x.dog(x)$
- Nom_3 (dog in the park), as PP.sem(Nom_2.sem):

$$\begin{aligned}
&(\lambda Q \lambda x.\exists! z.park(z) \wedge in(x,z) \wedge Q(x))[\lambda y.dog(y)] && (26)\\
&= \lambda x.\exists! z.park(z) \wedge in(x,z) \wedge (\lambda y.dog(y))[x] && (27)\\
&= \lambda x.\exists! z.park(z) \wedge in(x,z) \wedge dog(x) && (28)
\end{aligned}$$

- a, Det_2: $\lambda P.\lambda Q.\exists x.P(x) \wedge Q(x)$
- NP_2 (a dog in the park), as Det_2.sem(Nom_3.sem):

$$\begin{aligned}
&(\lambda P.\lambda Q.\exists x.P(x) \wedge Q(x))[\lambda x.\exists! z.park(z) \wedge in(x,z) \wedge dog(x)] && (29)\\
&= (\lambda P.\lambda Q.\exists x.P(x) \wedge Q(x))[\lambda y.\exists! z.park(z) \wedge in(y,z) \wedge dog(y)] && (30)\\
&= \lambda Q.\exists x.(\lambda y.\exists! z.park(z) \wedge in(y,z) \wedge dog(y))[x] \wedge Q(x) && (31)\\
&= \lambda Q.\exists x.\exists! z.park(z) \wedge in(x,z) \wedge dog(x) \wedge Q(x) && (32)
\end{aligned}$$

- sees, Vt: $\lambda P \lambda x.P(\lambda y.\exists e.seeing(e) \wedge seer(e,x) \wedge seen(e,y))$

- VP, as Vt.sem(NP$_2$.sem):

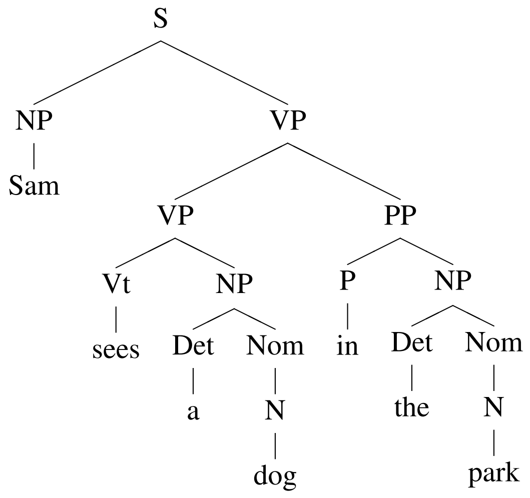
$$\begin{aligned}
&(\lambda P \lambda x.P(\lambda y.\exists e.seeing(e) \land seer(e,x) \land seen(e,y)))[\lambda Q.\exists w.\exists !z.park(z) \land in(w,z) \land dog(w) \land Q(w)] \\
&= \ldots \\
&= \lambda x.\exists w.\exists !z.park(z) \land in(w,z) \land dog(w) \land \exists e.seeing(e) \land seer(e,x) \land seen(e,w)
\end{aligned}$$

- Sam, NP$_3$: $\lambda P.P(Sam)$
- And finally, if we apply NP$_3$.sem(VP.sem), we can reduce the entire MR to:

$$\exists w.\exists !z.park(z) \land in(w,z) \land dog(w) \land \exists e.seeing(e) \land seer(e,Sam) \land seen(e,w)$$

The derivation is tedious, but it is nice to see that the resulting MR is what we'd expect: the dog is in the park, and Sam sees the dog.

I will not go through the whole derivation for the second tree, where PP attaches to VP:



Suffice to say that it derives the following MR:

$$\exists !z.park(z) \land in(Sam,z) \land \exists w.dog(w) \land \exists e.seeing(e) \land seer(e,Sam) \land seen(e,w)$$

This is not too bad for the current sentence: Sam is in the park and sees the dog. However, it's only accidentally right: technically the seeing event is the thing that should be in the park, not Sam. For "see" it's hard to separate the two, but consider the syntactically identical sentence "Sam cooks a potato in the oven". In this case it's clear that the cooking, not Sam, is the thing "in the oven".

So the correct meaning should really be:

$$\exists !z.park(z) \land \exists w.dog(w) \land \exists e.seeing(e) \land seer(e,Sam) \land seen(e,w) \land in(e,z)$$