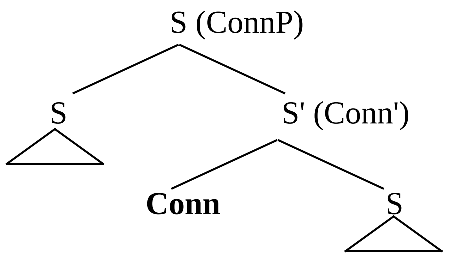
Semantics 564 **Lecture 14** Oct. 12, 1999

Homework:

1. P. 31 Exercise 2.

i) Revised phrase structure trees for "and" and "or":

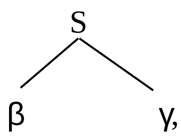


ii) Revised lexical entries for "and" and "or"

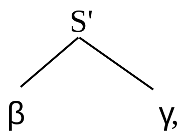
[[**and**]] = f: {0,1} --> {g: g is a function from {0,1} --> {0,1}}
For all x {0,1}, f(x) = g_x: {0,1} --> {0,1}
For all y {0,1}, g_x(y) = 1 iff x=1 and y=1

[[**or**]] = f: {0,1} --> {g: g is a function from {0,1} --> {0,1}}
For all x {0,1}, f(x) = g_x: {0,1} --> {0,1}
For all y {0,1}, g_x(y) = 1 iff x=1 or y=1

iii) Revised interpretation rules for S:



If α has the form β γ, then [[α]] = [[γ]]([[β]])



If α has the form β γ, then [[α]] = [[β]]([[γ]])

iv) [[S]] D_t
[[S']] $D_{<t,t>}$
[[**Conn**]] $D_{<t,<t,t>>}$

2. P. 40 Exercise 3.

(=1ii in λ-notation)

$[\lambda x: x \quad D_t . [\lambda y: y \quad D_t . x=1 \text{ and } y=1]]$

$[\lambda x: x \quad D_t . [\lambda y: y \quad D_t . x=1 \text{ or } y=1]]$

3. p. 32 Exercise 3 (this is *looong*. Do it thoroughly, though, in particular, be pedantic about (c). It'll pay off later, honest. Also, when you give the new lexical entry for **introduce**, give it in both λ-notation **and** the longer, easier-to-understand notation of the type in 22 above).

(a)

(i) New lexical entries:

[[**Maria**]] = Maria

[[**Jacob**]] = Jacob

[[**introduce**]] = f: D_e --> {g: g is a function from D_e to $D_{<e,t>}$}

For all x D_e, f(x) = g_x : D_e --> {h: h is a function from D_e to D_t}

For all y D_e, $g_x(y) = h_y$: D_e --> D_t

For all z D_e, $h_y(z)$ = 1 iff z introduces x to y

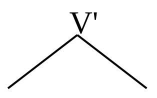
in lambda-notation

f:$[\lambda x: x \quad D_e . [\lambda y: y \quad D_e . [\lambda z: z \quad D_e$. z introduces y to x]]]

(ii) New semantic type for ditransitive verbs:

$D_{<e, <e, <e, t>>>}$

(iii) New structural interpretation rule, in addition to the new one for PP that H&K give:

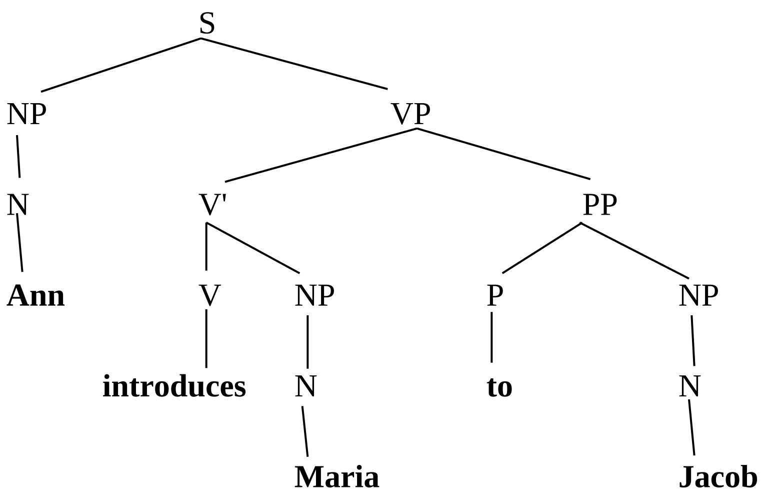


If α has the form β γ, then [[α]] = [[β]]([[γ]])

(b) Table displaying the actual value of the function denoted by **introduce** in a world where Ann introduces Maria to Jacob and Maria introduces Jacob to Ann:

$$
f: \quad \left(\begin{array}{ll}
\text{Ann -->} & \left(\begin{array}{lll}
\text{Ann} & \text{-->} & \left(\begin{array}{l}\text{Ann--> 0}\\ \text{Maria-->0}\\ \text{Jacob-->0}\end{array}\right)\\
\text{Maria} & \text{-->} & \left(\begin{array}{l}\text{Ann-->0}\\ \text{Maria-->0}\\ \text{Jacob-->0}\end{array}\right)\\
\text{Jacob} & \text{-->} & \left(\begin{array}{l}\text{Ann--> 0}\\ \text{Maria-->0}\\ \text{Jacob-->0}\end{array}\right)
\end{array}\right)\\
\text{Maria -->} & \left(\begin{array}{lll}
\text{Ann} & \text{-->} & \left(\begin{array}{l}\text{Ann-->0}\\ \text{Maria-->0}\\ \text{Jacob-->0}\end{array}\right)\\
\text{Maria} & \text{-->} & \left(\begin{array}{l}\text{Ann-->0}\\ \text{Maria-->0}\\ \text{Jacob-->0}\end{array}\right)\\
\text{Jacob} & \text{-->} & \left(\begin{array}{l}\text{Ann-->1}\\ \text{Maria-->0}\\ \text{Jacob-->0}\end{array}\right)
\end{array}\right)\\
\text{Jacob} & \left(\begin{array}{lll}
\text{Ann} & \text{-->} & \left(\begin{array}{l}\text{Ann-->0}\\ \text{Maria-->1}\\ \text{Jacob-->0}\end{array}\right)\\
\text{Maria} & \text{-->} & \left(\begin{array}{l}\text{Ann-->0}\\ \text{Maria-->0}\\ \text{Jacob-->0}\end{array}\right)\\
\text{Jacob} & \text{-->} & \left(\begin{array}{l}\text{Ann-->0}\\ \text{Maria-->0}\\ \text{Jacob-->0}\end{array}\right)
\end{array}\right)
\end{array}\right)
$$

(c) Interpret the given tree using the table just created:



(i) [[V]] = [[**introduce**]] by the following rule:

If α has the form $\begin{array}{c} \text{V} \\ | \\ \beta \end{array}$, then [[α]] = [[β]]

[[**introduce**]]=the whole table from (b) above, so [[V]] = the whole table from (b) above.

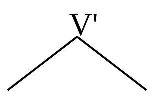
(ii) [[N]] = [[**Maria**]] = Maria by the following rule:

If α has the form $\begin{array}{c} \text{N} \\ | \\ \beta \end{array}$, then [[α]] = [[β]]

(iii) [[NP]] = [[N]] = Maria by the following rule:

If α has the form $\begin{array}{c} \text{NP} \\ | \\ \beta \end{array}$, then [[α]] = [[β]]

(iv) [[V']] = [[**introduces**]](Maria) by the following rule (which we just introduced):

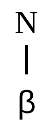


If α has the form β γ, then [[α]] = [[β]]([[γ]])

The value of [[**introduces**]](Maria) and hence of [[V']] is:

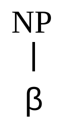
$$f: \left(\begin{array}{lll} \text{Ann} & \text{-->} & \left(\begin{array}{l} \text{Ann-->0} \\ \text{Maria-->0} \\ \text{Jacob-->0} \end{array} \right) \\ \\ \text{Maria} & \text{-->} & \left(\begin{array}{l} \text{Ann-->0} \\ \text{Maria-->0} \\ \text{Jacob-->0} \end{array} \right) \\ \\ \text{Jacob} & \text{-->} & \left(\begin{array}{l} \text{Ann-->1} \\ \text{Maria-->0} \\ \text{Jacob-->0} \end{array} \right) \end{array} \right)$$

(v) [[N]] = [[**Jacob**]] = Jacob by the following rule:

If α has the form 

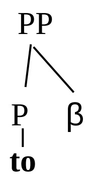
, then [[α]] = [[β]]

(vi) [[NP]] = [[N]] = Jacob by the following rule:

If α has the form 

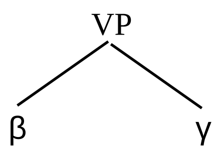
, then [[α]] = [[β]]

(vii) [[PP]] = [[NP]] = Jacob by the following rule (given by H&K):

If α has the form 

, then [[α]] = [[β]]

(viii) VP = [[V']](Jacob) = [[**introduce**]](Maria)(Jacob) by the following rule:

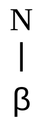
If α has the form 

, then [[α]] = [[β]]([[γ]])

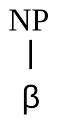
The value of [[**introduce**]](Maria)(Jacob) and hence of [[VP]] is:

$$f: \quad \begin{pmatrix} \text{Ann-->1} \\ \text{Maria-->0} \\ \text{Jacob-->0} \end{pmatrix}$$

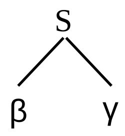
(ix) [[N]] = [[**Ann**]] = Ann by the following rule:

If α has the form  , then [[α]] = [[β]]

(x) [[NP]] = [[N]] = Ann by the following rule:

If α has the form  , then [[α]] = [[β]]

(xi) [[S]] = [[VP]](Ann) = [[**introduce**]](Maria)(Jacob)(Ann) by the following rule:

If α has the form  , then [[α]] = [[γ]] ([[β]])

According to the table, the value of [[**introduce**]](Maria)(Jacob)(Ann) = 1.

(d) For any x, y, z D_e, f(x)(y)(z) =1 iff z introduces x to y.

4. p. 39. Exercise 1.

(a) This is a function that gives the value 1 for any natural number provided that number is greater than three and less than seven. (It's a function from numbers to truth values; it's the characteristic function of the set {4,5,6}).

(b) This is a function that takes people as its argument and gives as its value the father of the person that is the argument -- it maps every person to his/her father.

(c) This function takes any set X in the power set of D, and gives as its value the set of all the members of D that are not in X, i.e. the complement of X (that is, D-X).

(d) This function takes any set X that is a subset of D and maps it onto a function that is true of any element of D that is not a member of X, i.e. the characteristic function of the complement of X.

564 Lecture 14 Oct. 12, 1999

1 Reducing the inventory of interpretation rules

So far, all our interpretation rules fall into two classes. The first class tells us how to interpret our lexical items, i.e. the terminal nodes: go to the lexicon and look them up. The second class tells us how to interpret subtrees of syntactic structure: if it's an NP dominating a single node, its interpretation is such-and-such, if it's an S dominating a branching node, its interpretation is such-and-such. Because you are astute proto-linguists, you will have noticed that there's a generalization to be made about all these rules. If a node is non-branching, it inherits the denotation of its daughter node, and if it branches, one of the branches is a function that takes the other of the branches as its argument, and the denotation of the whole is the value of the function at that argument. So rather than including rules for branching S nodes, branching VP nodes, branching V' nodes, etc., we can write just one rule that applies to any branching node; similarly for the non-branching nodes. Here they are (H&K's (2) and (3) p.44):

1. *Non-branching nodes (NN)*
   If α is a non-branching node, and β is its daughter, then $[[\alpha]]=[[\beta]]$

2. *Functional Application (FA)*
   If α is a branching node, $\{\beta,\gamma\}$ the set of α's daughters, and $[[\beta]]$ is a function whose domain contains $[[\gamma]]$, then $[[\alpha]] = [[\beta]]([[\gamma]])$.

And of course we've got our lexical items interpretation:

3. *Terminal Nodes (TN)*
   If α is a terminal node, then $[[\alpha]]$ is specified in the lexicon

(The abbreviations TN, FA and NN are for use in specifying which rule you use at a particular step in a proof, so you don't have to say, "by Functional Application" - like our abbreviations M.P. for Modus Ponens in earlier proofs).

As H&K note, this is called "type-driven interpretation" because so far it doesn't refer to any particular syntactic structures, only to the types of the constituents involved:

basically, in some sense, the types of the constituents decide for themselves how they will combine.

2 Binary branching and conjunction

It's further worth noting that in this interpretation structure we can't interpret ternary branching (or n-ary more than binary branching) structures, like those for *and* that we developed last week. So we're stuck with assuming binary-branching structures for connectives. Of course, this is not an inherent limitation of the mechanics, but is rather a theoretical choice to limit the mechanics. We could just as easily have assumed the n-ary functional interpretation rule that we initially did for "and" above.

What are the consequences of this choice? One consequence is that the semantic theory fits neatly into the syntactic theory most adopted by H&K's colleagues (and ours), according to which the syntax has to be binary branching. However, at least for the case of the sentential connectives, Jim McCawley, e.g., has argued against the view that they are at most binary functions.

(This argument is from p. 88-89 of *Everything that Linguists have Always Wanted to Know About Logic**)

4. (a) Alice ordered pork chops, Ben ordered liver and Sylvia ordered lasagna.
   (b) Alice ordered pork chops and Ben ordered liver and Sylvia ordered lasagna
   (c) (S (and) (S and S))

If the iterated conjunction in 4a and b is necessarily binary branching, then 4c is necessarily its structure. In 4a, presumably, the first "and" is just deleted because it's redundant.

Now, consider a structural possibility that exists in English when the conjuncts are all of the same shape (contain the same verb) as in 4. above. It's called *Gapping*, and what happens is that you can include the verb in the first conjunct, and then just delete it in the second and third conjuncts, under identity (5).

5. *Gapping*
   Alice ordered pork chops, (and) Ben liver, and Sylvia lasagna.

Now, here is where it gets interesting. Compare Gapping in sentences with all the "ands" to the sentences without, in 6, below. When you don't have all the "ands", Gapping seems to be a so-called "Across-the-Board" phenomenon, because you have to do it for every element in the multiple conjunction. You can't choose to do it for just the first two elements, or just the last two, as in (6a) and (6b) below.

6. *Across-the-Board for 4a, but not for 4b*
   - a. ?Alice ordered pork chops, Ben liver, and Sylvia ordered lasagna.
   - b. ??Alice ordered pork chops, Ben ordered liver, and Sylvia lasagna.
   - c. Alice ordered pork chops and Ben liver, and Sylvia ordered lasagna.
   - d. Alice ordered pork chops, and Ben ordered liver and Sylvia lasagna.

(Aside: "Across-the-Board" was originally invented to describe the badness of the following:

Who did Alice love, Jill like, and Bill hate?
*Who did Alice love, Jill like, and Bill hated Sue?

Note that this badness is much badder than the badness in 6)

Now, let's assume that Gapping is a structural deletion rule, that applies to conjoined sentences. The nicest way to account for the badness of 6a and b is to say that all the Ss conjoined with each other at once, forming a (quaternary?) branching structure. Gapping then applies once, and has to apply to each of the conjuncts in the structure. 6c and d, however, can be binary branching, and Gapping can apply to any pair of sub-constituents. Allowing n-ary conjunction in cases like 4a can account for the difference between 6ab and 6cd.

Further, what about the following:

7. *Different verbs:*
   - a. *Alice ate a hamburger, Bill drank some beer and Sylvia a Coke.
   - b. Alice ate a hamburger, and Bill drank some beer and Sylvia a Coke.
   - c. ??Alice ate a hot dog, Bill a hot dog, and Sylvia drank a Coke.

8. *Different factoring*:
   - a. Alice wrote about quasars, Ted wrote about black holes, and Oscar wrote about supernovas.
   - b. Alice wrote about quasars, Tom black holes and Oscar supernovas.
   - c. Alice wrote about quasars, Tom about black holes, and Oscar about supernovas.

d. *Alice wrote about quasars, Tom black holes, and Oscar about supernovas.
e. *Alice wrote about quasars, Tom about black holes, and Oscar supernovas.

So, McCawley concludes, conjunction must be permitted to be of n-ary branchingness, and hence H&K's interpretation function isn't going to quite cut it in these cases. It's a fairly minor point, though, and binary branching will work for 99% of structures, so let's not worry about it yet. (Wonder what Kayne et al. think of all this? Probably they make an exception for conjunction, or dispute the judgments here).

Ok, back to H&K. They note that since none of their nodes in the interpretation rules bear category labels, they're appropriate for syntaxes that don't have category labels at all. Modern minimalism is a theory of this type. However, it *does* entail that there's a single tree that is the input to the semantic interpretation. Using both DS and SS trees won't work, for example, nor will using both PF and LF representations as input.

3 Interpretability: semantic well-formedness

H&K point out that lots of interpretable sentences, according to their rules, might turn out to be syntactically ill-formed. So, e.g. speakers of English would agree that 9a is ungrammatical, although it clearly is interpretable, and similarly for 9b:

9. a. Me ape-man.
   b. Tarzan love Jane.

So there are syntactic rules that rule out certain structures independently of the semantics. Are there structures that are acceptable to the syntax but ruled out by the semantics that we've got so far? What about Chomsky's famous example:

10. Colorless green ideas sleep furiously.

He claimed that this was syntactically perfectly well-formed, but semantically garbage.

Now, actually, that's not quite fair. As far as the interpretation goes, it's also perfectly well formed; that is, every element in the tree is functionally applying to another element that is a suitable argument, where appropriate. We can see this if we replace the actual lexical items with equivalent members of the same class:

11. Heartless evil tyrants sleep soundly.

11 makes perfectly good sense, so it's not that 10 is semantically *ill-formed* (i.e. ungrammatical), it's just that the interpretation the sentence gets doesn't make any sense because of what we know about the world. H&K want to point out that on the semantics we've worked out so far, it might well be the case that a certain structure will be syntactically grammatical, but semantically ungrammatical. They give as an example:

12. *Ann laughed Jan.

What if we permitted the syntactic component to generate this, with a transitive verb's structure? What would happen when it got to the semantics? Well, the VP node would represent the functional application of [[**laugh**]] to [[**Jan**]], which would give the VP the denotation of 1 iff Jan laughed, and then would break down when it came time to interpret the S node, because none of the interpretation rules will work. Then, 12 is not ungrammatical in the syntactic sense, but rather just uninterpretable. They make this more explicit by rewriting the interpretation rules as specifying whether or note a node is *in the domain of the interpretation function*. Then, they propose a semantic principle that will outlaw uninterpretable trees:

13. *Principle of Interpretability*
All nodes in a phrase structure tree must be in the domain of the interpretation function [[ ]].

Now, as things stand, the syntactic component doesn't generate structures like 12. The Theta-Criterion rules them out. The theta-criterion is the following:

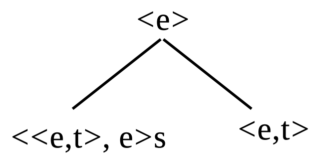
14. θ-Criterion
Each argument bears one and only one θ-role, and each θ-role is assigned to one and only one argument.

15. θ-grid for "love"
love: [(Lover), Lovee]
([(Agent), Theme])

Now, the theory says that in the lexicon, verbs have θ-grids, that is, lists of the θ-roles they have available to assign. "Love", e.g., has two theta-roles, "Lover" and "Lovee". (θ-roles are also often grouped into larger classes, like Agent, Patient, etc. -- but for the purposes of the θ-criterion, it's the particular roles of any given verb that count). When the verb is projected into the syntax, it assigns its theta-roles to its arguments, and if there's one more argument than theta-role, or one more theta-role than argument, there's a problem: a violation of the θ-criterion. The reason that 12 is bad, in regular GB, is this type of problem. H&K point out that in fact, if interpretation works the way we have been talking about, then most of the work of the θ-criterion could be done by the Principle of Interpretability, and we'd get rid of a lot of redundancy in the system.

Now, H&K go on to ask the question, which is stronger, the θ-criterion or the Principle of Interpretability? In fact, it's the θ-criterion, because for the θ-criterion, there must be an argument for every verb. In the case of the Principle of Interpretability, however, that's not so. What if we had a node in a structure that denoted a function of type <e,t>, as in 16. The θ-criterion would say that it has to assign its θ-role to an argument. All the Principle of Interpretability says is that the whole structure has to be interpretable. So, what if, rather than taking an argument, the function of type <e,t> is itself actually *taken* as an argument, by some other function, say, of type <<e,t>,e>.

16.



They argue that this is exactly the situation in 17b. In 17a, they argue, the denotation of "house" has to be a function of type <e,t>, just like "laugh" or other intransitive verbs, because 17a is true iff [[**Graceland**]] is a house, just like "Jan laughs" is true iff [[**Jan**]] laughs. So if "house" is of type <e,t>, then it takes an argument of type e, and according to the θ-criterion, it should have a θ-role to assign. But in 17b, there is no phrase for "house" to assign its θ-role to. Rather, they say that in fact what is going on is that the determiner is taking a function of type <e,t> as its argument.

17. a) Graceland is a house.
    b) The house burned down.

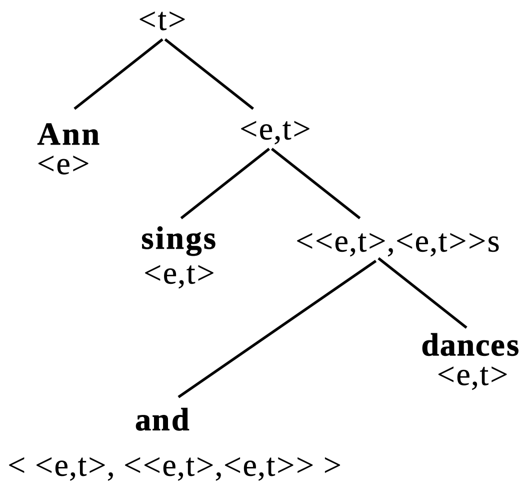
A more straightforward case is in 18.

18. Ann sings and dances.

This is a prima facie violation of the θ-criterion, because the single argument "Ann" gets two theta-roles, one from "sings" and one from "dances". However, it's easy enough to make it satisfy the principle of interpretability, once we have an appropriate denotation for "and":

19. $[[\textbf{and}_{\textbf{VP}}]] = [\lambda f \quad D_{<e,t>} . [\lambda g \quad D_{<e,t>} . [\lambda x \quad D_e . f(x)=g(x)=1]]]$

20.



So, H&K note, it's probably preferable to replace the θ-criterion with something like the Principle of Interpretability, and get rid of θ-grids altogether, eliminating a bunch of redundancy in the lexicon.