

Notes on Session 9 (22.12.2008)

In a somewhat minimal constellation, we read (the absent) Andreas' paper on Anti-Symmetrie. Even though the paper is work-in-progress, I think we got at least the basic idea(s) and I am quite sure that Andreas will be able to clarify all those unanswered questions we stumbled upon. Besides trying to put these questions into an answerable form, I'll also try to repeat my (hopefully not too confused) 'remarks' on relations.

As for the motivation for Anti-Symmetrie, just read Andreas' paper. In a nutshell, Anti-Symmetrie is supposed to answer the question what the exact relation between the hierarchical and the linear structure of a sentence is (how do the trees we always draw relate to the sounds we always emit). Remember, for example, that Merge is supposed to produce sets. The problem with this is that you do not know the linear order of the tree's leaves since the preterminal syntactic object (*I, know*) could be spelled out as both "I know" and "know I".

Linearization is supposed to do the trick, and this is where anti-symmetrie comes in. Let's have a look at the

Linear Correspondence Axiom: $d(A)$ is a linear ordering of T .

Now what is that supposed to mean? First of all, T stands for any phrase-marker (that is, if I am not completely mistaken, an arbitrary node). A *linear ordering* is binary relation, whereas this should not worry you since relations are nothing one should worry about and in the end, a linear order is a linear order, no matter how obscure one tries to define it.

Relations are subsets of cartesian products of sets. That is actually pretty easy to understand once one knows what a cartesian product is. Given two sets A and B , the cartesian product of these two sets,

written as $A \times B$, is defined as the set of ordered pairs consisting of an element out of A and an element out of B . Take for example the two sets Human (that is, the set {Neven, Iwo, Mik, Andreas1, Andreas2, Richard, Benni, ...}) and Heights (that is, {..., 170cm, ..., 190cm, ...}). The cartesian product $\text{Human} \times \text{Heights}$ is the set $C = \{ \langle \text{Neven}, 1\text{cm} \rangle, \langle \text{Neven}, 2\text{cm} \rangle, \dots, \langle \text{Neven}, 500\text{cm} \rangle, \dots, \langle \text{Iwo}, 1\text{cm} \rangle, \dots \}$. You get the idea, I hope – just pair any element out of A with any element out of B . It is obvious that C can get no real good interpretation. What is it supposed to mean that there is an element $\langle \text{Neven}, 1\text{cm} \rangle$ (for it is certain that Neven is not 1cm big)? Therefore, relations are SUBSETS of cartesian products. Take for example the relation 'has_height'. This is a binary (for it relates two 'entities') relation between humans and heights. It's obvious that this is a subset of C , since it should look something like the following: $\{ \langle \text{Neven}, 182\text{cm} \rangle, \langle \text{Mik}, 172\text{cm} \rangle, \langle \text{Benni}, 178\text{cm} \rangle, \dots \}$ ¹, and this is a subset of C (since $\langle \text{Neven}, 182\text{cm} \rangle$ and all the other elements are also in C). Given a relation R (that is, the set), you can say that the *Relation* R (the concept you wanted to encode formally) holds between b and c iff $\langle b, c \rangle$ is in R (the set). Relations can have certain properties, such as transitivity. Transitivity simply means that whenever a is in relation R to b (written as aRb) and that bRc , then also aRc . Take for example 'bigger than' or 'equal' as typical transitive relations. If $5 > 2$ and $20 > 5$, then $20 > 2$. And if 'The US president' = 'G.W.Bush' and 'G.W.Bush' = 'The biggest moron alive', then 'The US president' = 'The biggest moron alive'.² A linear order of a set is also a binary relation between the set and itself. If we call our set A , the linear order is a subset of $A \times A$. Think for example

¹ Except for my own height, those numbers are more or less guessed.

² Excuse my sloppy use of quotation marks, and of course there's more to the above example since we're talking descriptions on the one hand and a proper name on the other, but we are not in a philosophy reading circle :)

of the relation 'bigger_than_or_as_big_as' defined on humans. Then we have something like {<Iwo, Iwo>, <Iwo, Richard>, ...}. If a relation is a linear order, it has certain properties, namely Transitivity (we had that), Totality and Antisymmetrie (not the kind we are interested in, though). A relation is Total iff for any a and b, there is <a,b> or <b,a> (or both). The relation 'loves', for example, is not total, for there are quite a lot of people such that neither loves the other. The relations 'as_big_as' and 'bigger_than' is also not total, whilst the relation 'bigger_than_or_as_big_as' is. The linear ordering has to be total, since we want to know for every leave whether it precedes another leave or not and it is total, since every word either precedes another word in a sentence or it does not.

Antisymmetrie is a somewhat obscure property – it states that whenever you have two elements such that both aRb and bRa, it follows that a=b (so you do not really have two elements a and b, but only one...). This is important for a linear order since it should not be the case that a precedes b and b precedes a (it should also not be the case that a precedes a, but this is another property, namely anti-reflexivity).

Let's apply the new notions to syntax. We have

D, the dominance relation between nonterminals (that is, something like {<CP, IP>, <CP, VP>, <CP, NP>, ..., <NP, N'>, ...})

d, the dominance relation from nonterminals to terminals (something like {<NP₁, peter>, <VP, kill>, <VP, mary>, <NP₂, mary>, ...})

the FUNCTION d(X), that yields, for example, d(VP) = {kill, mary}; it yields the set of all terminals dominated by X

d(<X,Y>) then yields d(X)d(Y) – this might need an example. Take d as above, then d(NP₁)={peter} and d(VP)={kill, mary}. Then

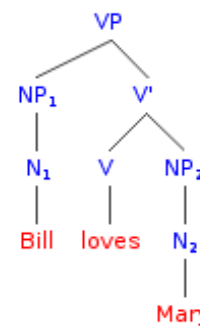
d(<NP₁, VP>) = {<peter, kill>, <peter, mary>}

Now all that we need is to define the set A – let A be the set consisting

of all pairs of non-terminals such that the first element ASYMMETRICALLY c-commands the second.

Asymmetric c-command is pretty straightforward: a asymmetrically c-commands b iff a c-commands b and b does not c-command a.

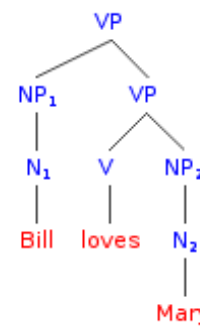
Some examples:



A is the set {<NP₁, V>, <NP₁, NP₂>, <NP₁, N₂>, <V', N₁>, <V, N₂>}
 d(A) = {<Bill, loves>, <Bill, Mary>, <loves, Bill>, <Mary, Bill>, <loves, Mary>}

Now this obviously DOES NOT yield the right result.³ d(A) is not a linear ordering, since the requirement of anti-symmetrie is not satisfied. We have both <Bill, loves> and <loves, Bill>, and <Bill, Mary> and <Mary, Bill>. This

problem is dealt with by the assumption that specifiers are adjuncts. Consider the tree: All that has changed is that there is no longer a V' level. Instead, there is a *two segment* VP category. Now both domination and c-command are notions that are applied to categories – in order for a category to c-command another, every segment of it needs to c-command. This has as a consequence, that A now is the set {<NP₁, V>, <NP₁, NP₂>, <NP₁, N₂>, <V, N₂>} - note that <VP, N₁> (or, in the old tree, <V', N₁>) is no longer in A, since the upper-segment of VP does not c-command N₁ (for it dominates N₁). Now,



³ I hope, however, that this made clear how the notions are to be applied, that is, what A and d(A) mean in a concrete example.

$d(A)$ yields: {<Bill, loves>, <Bill, Mary>, <loves, Mary>}, and this is a linear ordering of the set {Bill, Mary, loves}. You can easily check this by looking at the three requirements:

Totality: Every element stands in the relation to the other.

Transitivity: We have <Bill, loves>, <loves, Mary> and <Bill, Mary>

Antisymmetrie: We changed the tree in order to fulfill this requirement (see above).

Somewhat condensend, but that is the basic idea behind anti-symmetrie, as far as I grasped it.

To summarize: We need anti-symmetric syntactic structures in order to get a linear mapping from our hierarchical tree via the notion of asymmetric c-command and the function $d(X)$. Therefore, we need to do away with specifiers, treating them as adjuncts (now this is one of the points that need clarification – is that right? and if it is, can we still distinguish specifiers from mere adjuncts and talk about something like the specifier-head-relation?).

The other questions that need an answer:

- why is it that right-adjunction would violate the ECP (empty category principle = every empty category, these including traces, needs to be properly governed or something like this) (cf. p.2)? We tried to make some sense of it, and for right-movement it was more or less intuitively clear. But how does right-adjunction relate to empty categories??
- Why should the formulation of c-command on p.4 pose a problem for root contexts? The formulation was:
 X c-commands Y iff X and Y are categories and X excludes Y and every category that dominates X dominates Y .
I bolded (?) the part that to me seems to relate to the worries about root contexts. This is, however, a conditional statement.

IF there are categories dominating X , THEN these must also dominate Y . In root contexts, there simply are no categories dominating X – this, however, does not have as a consequence that there is no c-command relation but simply renders the last part of the definition irrelevant for those cases. Perhaps there is something we missed, so the last question is: Why worry about root contexts?