564 Lecture 16 Oct. 19, 1999

1 Homework

1. Characteristic functions of the sets making up the denotation of R_{adores} and $R_{assigns\ to}$:

a) f_{adores}

<Jacob, Jacob> -->	0
<Jacob, Maria> -->	1
<Maria, Maria> -->	1
<Maria, Jacob> -->	0

(b) $f_{assigns\ to}$

<Jacob, Jacob, Jacob> -->	0
<Jacob, Jacob, Maria> -->	1
<Jacob, Maria, Jacob> -->	0
<Jacob, Maria, Maria> -->	0
<Maria, Maria, Maria> -->	0
<Maria, Maria, Jacob> -->	0
<Maria, Jacob, Maria> -->	1
<Maria, Jacob, Jacob> -->	0

2. Schönfinkeled versions, right-to-left:

(a)

Jacob -->	Jacob -->	0
	Maria -->	0

Maria -->	Jacob -->	1
	Maria -->	1

(b)

Jacob -->	Jacob -->	Jacob -->	0
		Maria -->	0
	Maria -->	Jacob -->	0
		Maria -->	0
Maria -->	Jacob -->	Jacob -->	1
		Maria -->	1
	Jacob -->	Jacob -->	0
		Maria -->	0

3. (a) The right-to-left Schönfinkelization of f_{adores} is indeed a suitable denotation for the English verb. The verb takes its object (the rightmost argument in the ordered pair of the set-theoretic representation) as its first argument, and maps that to a function which takes its subject (the leftmost argument in the pair) as its argument and maps that to a truth-value.

(b) The right-to-left Schönfinkelization of $f_{assigns\ to}$ is *not* a suitable denotation for the English verb. Assuming a syntactic tree for "Jacob assigns Jacob to Maria" that is the same as the tree for **introduce**, then the verb does not combine with the rightmost argument in the set-theoretic ordered triple first. Rather, it takes the intermediate argument as its direct object, then the rightmost argument as the object of *to*, and finally the leftmost argument.as its subject.

4.

(a) $[\lambda x \quad D\,.\,[\lambda y \quad D\,.\,[\lambda z \quad D\,.\,z \text{ introduced } x \text{ to } y]]](\text{Ann})(\text{Sue}) =$

$[\lambda y \quad D\,.\,[\lambda z \quad D\,.\,z \text{ introduced Ann to } y]](\text{Sue}) =$

$[\lambda z \quad D\,.\,z \text{ introduced Ann to Sue}]$

(b) $[\lambda x \quad D\,.\,[\lambda y \quad D\,.\,[\lambda z \quad D\,.\,z \text{ introduced } x \text{ to } y](\text{Ann})](\text{Sue})] =$

$[\lambda x \quad D\,.\,[\lambda z \quad D\,.\,z \text{ introduced } x \text{ to Sue}](\text{Ann})] =$

$[\lambda x \quad D\,.\,\text{Ann introduced } x \text{ to Sue}]$

(c) $[\lambda x \quad D\,.\,[\lambda y \quad D\,.\,[\lambda z \quad D\,.\,z \text{ introduced } x \text{ to } y](Ann)]](Sue) =$
$[\lambda y \quad D\,.\,[\lambda z \quad D\,.\,z \text{ introduced Sue to } y](Ann)] =$
$[\lambda y \quad D\,.\,\text{Ann introduced Sue to } y]$

(d) $[\lambda x \quad D\,.\,[\lambda y \quad D\,.\,[\lambda z \quad D\,.\,z \text{ introduced } x \text{ to } y]](Ann)](Sue) =$
$[\lambda y \quad D\,.\,[\lambda z \quad D\,.\,z \text{ introduced Sue to } y]](Ann) =$
$[\lambda z \quad D\,.\,z \text{ introduced Sue to Ann}]$

(e) $[\lambda f \quad D_{<e,t>}\,.\,[\lambda x \quad D\,.\,f(x) = 1 \text{ and } x \text{ is gray}]]([\lambda y \quad D\,.\,y \text{ is a cat}]) =$
$[\lambda x \quad D\,.\,[\lambda y \quad D\,.\,y \text{ is a cat}](x) = 1 \text{ and } x \text{ is gray}] =$
$[\lambda x \quad D\,.\,x \text{ is a cat and } x \text{ is gray}]$

(f) $[\lambda f \quad D_{<e,<e,t>>}\,.\,[\lambda x \quad D\,.\,f(x)(Ann)=1]]([\lambda y \quad D.[\lambda z \quad D.z \text{ saw } y]])=$
$[\lambda x \quad D\,.\,[\lambda y \quad D.[\lambda z \quad D.z \text{ saw } y]](x)(Ann)=1] =$
$[\lambda x \quad D\,.\,[\lambda z \quad D.z \text{ saw } x](Ann)=1] =$
$[\lambda x \quad D\,.\,\text{Ann saw } x]$

(g) $[\lambda x \quad |N\,.\,[\lambda y \quad |N\,.\,y > 3 \text{ and } y < 7](x)] =$
$[\lambda x \quad |N\,.\,x > 3 \text{ and } x < 7]$

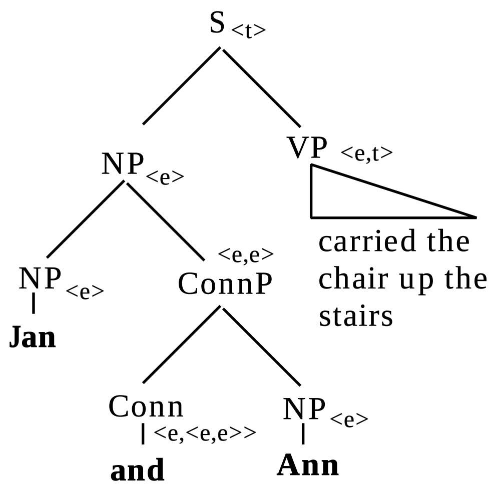
(h) $[\lambda z \quad |N\,.[\lambda y \quad |N\,.\,[\lambda x \quad |N\,.\,x > 3 \text{ and } x < 7](y)](z)] =$
$[\lambda z \quad |N\,.[\lambda y \quad |N\,.\,y > 3 \text{ and } y < 7](z)] =$
$[\lambda z \quad |N\,.\,z > 3 \text{ and } z < 7]$

or

$[\lambda z \quad |N\,.[\lambda y \quad |N\,.\,[\lambda x \quad |N\,.\,x > 3 \text{ and } x < 7](y)](z)] =$
$[\lambda z \quad |N\,.\,[\lambda x \quad |N\,.\,x > 3 \text{ and } x < 7](z)] =$
$[\lambda z \quad |N\,.\,z > 3 \text{ and } z < 7]$

3. Bonus question:

Drawing the (binary-branching) tree for "Jan and Ann carried the chair up the stairs" this is what we get (avoiding the detail of the structure of the VP), including the types that all the nodes must have:



We can see that **and** must be of type <e,<e,e>>, in order to make the type of the conjoined NP work out right. Here's a stab at the actual function:

[[**and**]]: [λx D **.** [λy D **.** the entity D that is a pair consisting of x and y]]

This has a major fudge in it. It is considering a "pair" an entity, an element of D_e, when really the (unordered) pair {x,y} is not a member of D_e but rather of $D_e x D_e$. But heck. The result is that Jan and Ann are necessarily carrying the chair up the stairs together, since it is the "pair" entity that is the subject argument of the VP. Once we get to figuring out DPs as (the characteristic functions of) sets of properties, we'll see how we can fix this.

2 Nonverbal predicates

So far, we're getting along with just our three rules. You'll have noticed that sometimes there's a branching node that gets interpreted as a non-branching node ([to Jan] in "Ann introduced Sue to Jan" is interpreted as [Jan]). There'll be a few more of these. One of them is "be".

4. (a) [John is rich].

(b) I consider [John rich].

(c) I consider [John to be rich].

Based on the equivalence of meaning of 4bc, we can for the moment assume that "be" makes no semantic contribution at all. The verb "to be", and "is" in our sentence 4a, will be treated as vacuous; for the moment, we'll interpret "John is rich" as "John rich", just like "John smokes". This means that a word like "rich" will have the same denotation as an intransitive verb, a function from <e,t>.

5. [[**rich**]] = [λx D_e **.** x is rich]

This is the characteristic function of the set of things that are rich.

Now, last time we had a vacuous node, we had a specific interpretation rule for it (the PP in the "introduce" tree). What we'll do here is just note that certain items are vacuous, and assume they don't count for semantic interpretation -- that is, a branching node containing a vacuous item on one branch is the same as a non-branching node, and gets the denotation of its non-vacuous daughter.

Now, remember in our predicate calculus we had to treat not only verbs and adjectives as predicates, but nouns as well; something like "All cats are mammals" was realized as "If x is a cat, then x is a mammal."

6. All cats are mammals = "If x is a cat, then x is a mammal"

x(Cat(x) --> Mammal(x)).

All cats play ="If x is a cat, then x plays."

x(Cat(x) --> Play(x))

We'll do the same thing in our lambda calculus here: we'll assume that nouns are the characteristic function of the set they denote, a function of type <e,t>, just like intransitive verbs and adjectives.

7. (a) Kaline is a cat.
   (b) I consider Kaline a cat.
   (c) I consider Kaline to be a cat.

It's clear that the same arguments that derived the vacuousness of "be" above, that "a cat" has to have the same denotation as "rich" or "smokes", that is, a function of type <e,t>. We'll assume that that's the denotation of "cat", and the "a" is vacuous in the same way that "be" is. (This won't do forever, but it'll be fine for now).

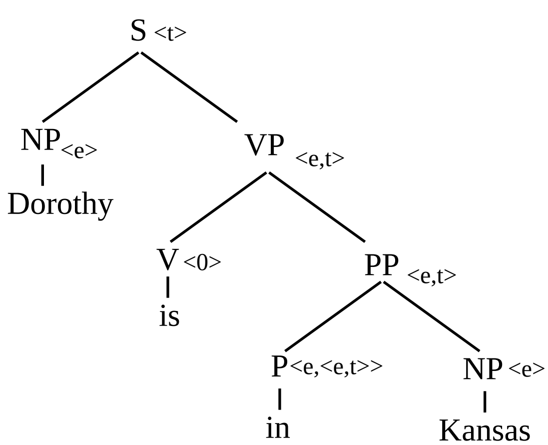
8. [[**cat**]] = [$\lambda x \quad D_e$ **.** x is a cat]

Similarly for a preposition like "out":

9. John is out.
   [[**out**]] = [$\lambda x \quad D_e$ **.** x is not at home]

Most prepositions are not intransitive, though, but rather transitive, and we'll treat them like we treated transitive verbs above, as functions of type <e,<e,t>>:

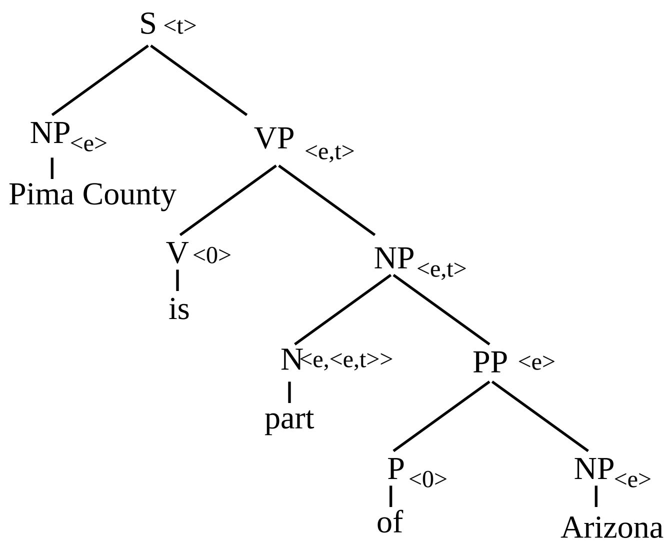
10. Dorothy is in Kansas.



$[[\textbf{in}]] = [\lambda x \quad D_e \,.\, [\lambda y \quad D_e \,.\, y \text{ is in } x]]$

There are transitive Ns and Adjs too, we'll treat them the same way:

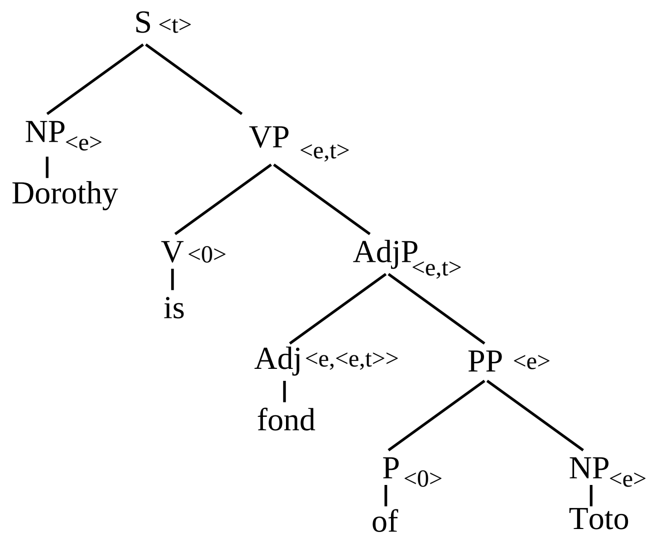
11. (a) Pima County is part of Arizona.



(Note that "of" is gong to be another vacuous node. This vacuity is in fact perfectly reasonable, and won't have to be altered except in one special case.)

$[[\textbf{part}]] = [\lambda x \quad D_e \,.\, [\lambda y \quad D_e \,.\, y \text{ is part of } x]]$

(b) Dorothy is fond of Toto.



12. H&K quote: "We will disregard the case of ditransitive (3-place, triadic) predicates, though there are presumably some analogs to verbs like "give" and "introduce" in other syntactic categories."

It's worth noting that nothing in this system nor the system before predicts the observed limitation of natural languages to predicates of no more than triadicity -- there are no natural language predicates that take 4 arguments necesarily. This restriction could possibly be of two types: it could be a by-product of the syntactic system, or it could be a processing problem -- human brains can't deal with more than 3 arguments. In fact, the former seems more likely than the latter, because we have no problem processing sentences like the following:

13. Dorothy sold the ruby slippers to Glenda for $1,000,000.

It'd be hard to imagine that there's some problem with making that "for $1,000,000" argument mandatory instead of optional. Also, the fact that I personally can't think of any triadic adjectives, nouns or prepositions makes me suspect that it's a by-product of the syntactic system -- there's something special about verbs that produces the possibility of an extra argument. (X-bar syntax means that you can only have 2 arguments in your immediate projection -- what if verbs have an extra head that allows them to take another argument?)

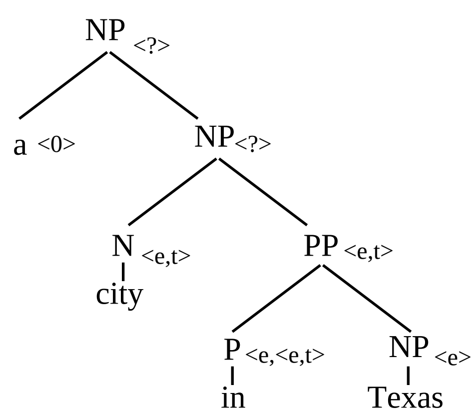
3 Restrictive modifiers.

14. (a) *PP as an argument of an NP*
a part of Europe
(b) *PP as a restrictive modifier*
a city in Texas
(c) *PP as a non-restrictive modifier*
Susan, from Nebraska

We won't worry about nonrestrictive modifiers; they're parenthetical comments the speaker makes in the process of asserting a separate proposition. H&K give as an example "It's suprising that Susan (from Nebraska) finds it cold in here."

Restrictive modifiers, on the other hand, we will worry about. Let's look at the tree for "a city in Texas."

15.



The problem is, how to compose the "city" node and the "PP" node, both of which are of type <e,t>, to get a denotation for the whole NP?

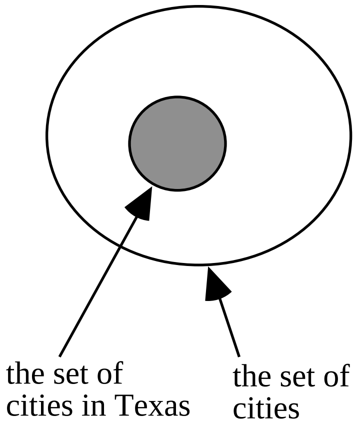
First, we can figure out what we want the denotation of the whole NP to be from looking at its behavior in copular sentences:

16. (a) Lubbock is [a city in Texas.]
    (b) I consider Lubbock [a city in Texas].

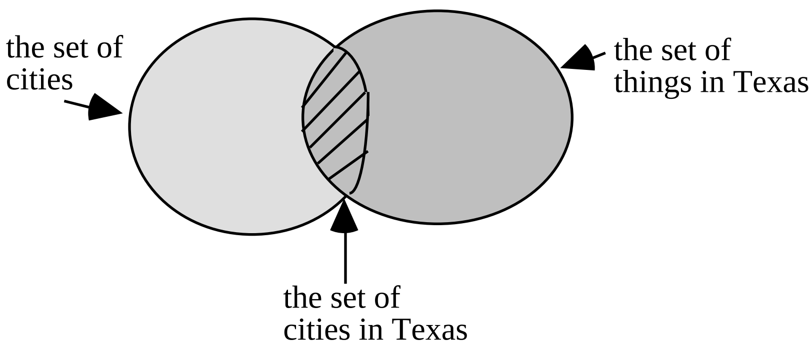
Here, we can see that the denotation of the whole NP also has to be a function from individuals to truth values, a function of type <e,t>. This makes sense, if we consider our intuitions about what "a city in Texas" denotes. "city", in our predicate calculus, denoted the set of cities, and "city in Texas" still denotes a set -- the set of cities in Texas, a subset of the set of cities, and a subset of the set of things in Texas: the intersection of the two sets.

17. Set-theoretic representations of "city in Texas":

(a)



(b)



So we want our NP node to denote the characteristic function of the set of cities in Texas. How can we acheive this?

4 First approach: new composition rule

Our current rule for composing the meanings of branching nodes, functional application, can't apply in this case. Neither of the two sister nodes (N, PP) are of an appropriate type to take the other as an argument, and functional application cannot apply. When functional application can't apply, we could try a different way of composing these two elements. They call it "Predicate Modification":

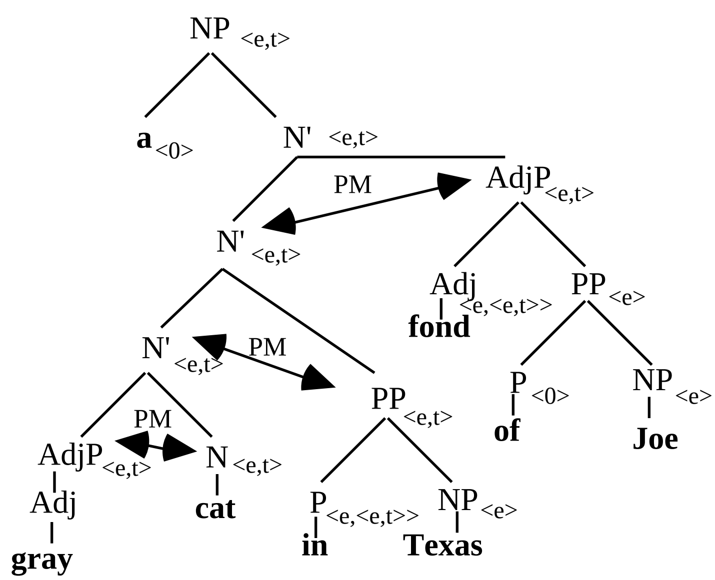
18. *Predicate Modification* (PM)

If α is a branching node, $\{\beta, \gamma\}$ the set of α's daughters, and $[[\beta]]$ and $[[\gamma]]$ are both of type <e,t>, then $[[\alpha]] = [\lambda x \quad D_e \,.\, [[\beta]](x)=[[\gamma]](x)=1]$

This does exactly what we want: it's the function that's true of all entities of which both its daughters are true. (Note this is exactly like the way that "and" functions to intersect predicates that we defined earlier.)

This will work not only for PP modifiers of nouns, but also adjectival modifiers or any number of stacked modfiers. H&K give the example sentence in 19:

19. Kaline is a gray cat in Texas fond of Joe.



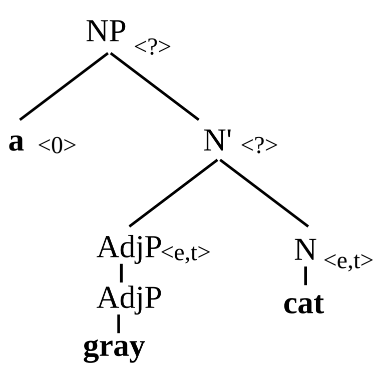
Now, if you're a syntactician, you'll notice that the syntactic rules have a lot to do with the legitimate shape structure, and the types of the elements involved don't have so much to do with it. Before, we had a nice neat system: predicates were of some type that took arguments, and arguments were of type <e>. We might have thought that we could get away with doing away with category labels altogether: if syntactic composition was functional application, we only had one legitimate way of putting together predicates with their arguments. But now we've got a rule that takes things of type <e,t> and combines them with other things of type <e,t>, irrespective of whether those things are nouns, adjectives, verbs, or whatever. Nothing about this structure says that the sentence "Kaline is a smoke in Texas cat fond of Joe" is ill-formed; we could make a binary-branching structure for that sentence that would get a perfectly legitimate interpretation. That sort of problem must, then, be syntactic, not semantic.

Spoiler: we're going to adopt this approach in the end. But first, we're going to consider another possible approach and see what it might entail, and some arguments pro and con. This will give you the feel for the kind of proposal that semanticists make and the kind of argumentation that they employ, as well as the (correct) impression that there's a lot more to say about modification.

5 **Second approach**: **modification as functional application and type-shifting**

Let's look at a modification tree again, and consider our problem from another angle:

20.



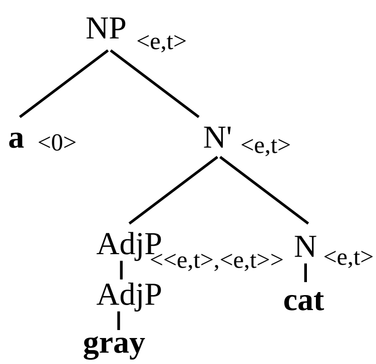
What if we insisted that "gray" and "cat" had to combine by functional application? One would have to be able to take the other as its argument. Let's say that we want the modifier to take the N as its argument. (Why should we do it this way as opposed to the other way? We'll answer this in a second, considering the case of stacked modifiers). We know we want the whole NP to be of type <e,t>. So **gray** will have to denote a function that takes a function of type <e,t> as its argument, and returns a function of type <e,t>. That is, it'll have to be a function of the type in 21a, with a meaning like that in 21b:

21. (a) $<<e,t>,<e,t>>$

(b) $[[\textbf{gray}]] = [\lambda f \quad D_{<e,t>} \textbf{ .} [\lambda x \quad D_{<e>} \textbf{ .} f(x) = 1 \text{ and } x \text{ is gray}]]$

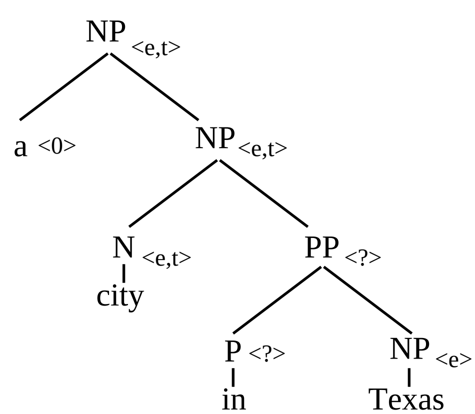
This'll give us the correct denotations, if we look at the way the tree looks:

22.



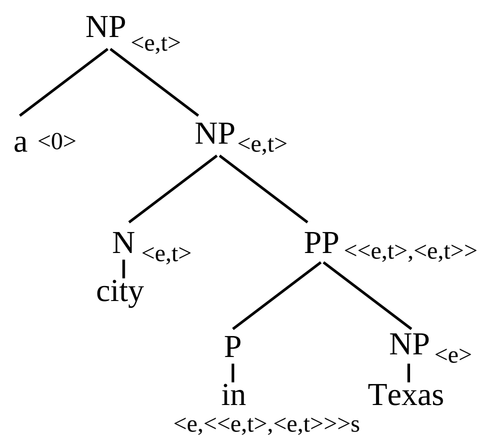
What about a diadic modifier, like the preposition **in**? Let's look at the tree for that:

23.



Now, **in** is going to have to be a function that takes individuals and returns a function that takes functions from individuals to truth values and returns a function from individuals to truth values. It's going to have to have the type in 24:

24. (a) <e, <<e,t>, <e,t>> >
(b)



(c) [[**in**]] = [λx D_e **.** [λf $D_{<e,t>}$ **.** [λy D_e **.** f(y)=1 and y is in x]]]

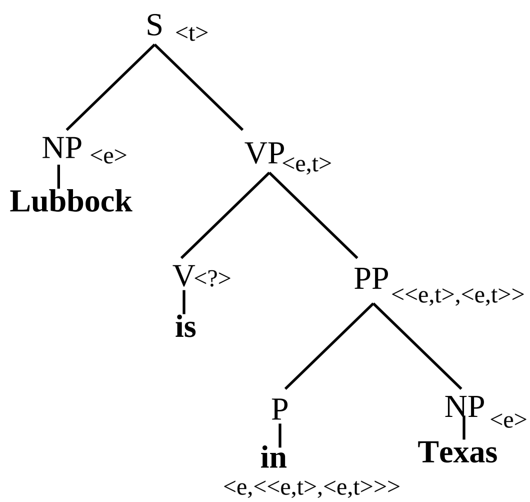
In this tree, f(y) =1 iff y is a city, so we have derived the correct truth conditions: "a a city in Texas" is a function that takes an argument (y) and returns the truth value 1 iff y is a city and y is in x, x=Texas. So if we can give modifiers like "in Texas" and "gray" appropriate types, we can retain the idea that all syntactic composition is functional application.

The trick is, though, that the types in 24a and 21a can't *always* be the types that **gray** and **in** have. Remember the original sentences?

25. (a) Kaline is gray.
(b) Lubbock is in Texas.

We're not about to change our denotations for proper names. Those are about the only things that we have a solid idea of what they mean. There's two possible ways we could go. We could revise the denotation for "be". Consider the following:

26.



The V and the PP have to combine via FA. Either the V has to take the PP as its argument, and return a function of type <e,t>, or the PP has to take the V as its argument, and return a function of type <e,t>. So the V (and hence **is**) will have to be either of two types:

27. (a) <<<e,t>,<e,t>>,<e,t>>
(b) <e,t>

Let's see what definition we'd need to make the first alternative work, where **is** takes the PP as its argument (half of H&K's 2nd exercise p 67):

28. $[[\textbf{is}]] = [\lambda f \quad D_{<<e,t>,<e,t>>} \,.\, [\lambda x \quad D_{<e>} \,.\, f(g)(x)=1$
$\text{when } g = [\lambda y \quad D_{<e>} \,.\, y \text{ exists}]]$

Other half of 2nd exercise: homework?

However, there's a number of problems with this proposal:

29. Problems with adapting the copula to combine with functions of type <<e,t>,<e,t>> :

(a) The copula will no longer combine with our type for nouns:
#Kaline is a cat.
(b) There are languages without a copula in this type of predication.

So **gray** and **in** have to have the <e,t> and <e,<e,t>> denotations that we already gave them. How do we reconcile this difference in their types for modification and predication?

Well, we could adopt the position that they've got *both* types. That is, anything of type <e,t> also has a type <<e,t>,<e,t>>, and anything of type <e,<e,t>> has a type <e,<<e,t>,<e,t>>. If we restrict this type-shifting to things labeled "Adj" and "P", then we also have a principled reason why Ns and Vs can't occur as modifiers without some special adaptation.

Next class: more on deciding between PM and type-shifting: nonintersective adjectives (**small**, **former**).

Distributivity and Conjunction

Dick Oehrle

University of Arizona Linguistics Colloquium
22 October 1999
2:00, Douglass 101

In arithmetic, multiplication is said to distribute over addition in virtue of the identity

a x (b + c) = (a x b) + (a x c).

The terminology is in fact applicable in a much wider range of settings. Thus, the set-theoretic operations of union (`\cup') and intersection (`\cap') distribute over each other, in view of the laws

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$
$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

In this paper, we investigate the distributivity of the complement of a conjoined expression over the conjunction.

As an illustration, consider the sentence

Smith gave Jones an LP and Peters a CD.

Abstractly, we may regard such a sentence as the result of conjoining `Jones an LP' with `Peters a CD' and combining the result of this conjunction with `Smith gave'. Calling the first of these operations conj and the second app, we can represent the sentence above as

(Smith gave) app ((Jones an LP) conj (Peters a CD)).

We can now ask whether app distributes over conj, a question that can be interpreted either with respect to syntactic well-formedness or to semantic interpretation. On the syntactic interpretation, we are interested in whether the syntactic well-formedness of undistributed sentences (such as Smith gave Jones an LP and Peters a CD) entails the syntactic well-formedness of the correponding distributed sentences (such as Smith gave Jones an LP and Smith gave Peters a CD). On the semantic interpretation, we are interested in whether the semantic properties of undistributed sentences (such as their truth-conditions) are identifiable with the corresponding properties of the corresponding distributed sentences.

In the paper, we show that distributivity of app over conj does not hold in general, either on the syntactic interpretation or on the semantic interpretation. For example, on the semantic side, if Smith owes Jones a nickel or Peters a dime, it need not be true that Smith owes Jones a nickel or Smith owes Peters a dime. And pairs such as the following illustrate the failure of syntactic distributivity:

Smith admires Jones's initiative and that she always turns chance to advantage.
. . . ?* Smith admires that she always turns chance to advantage.

As a consequence, theories of conjunction which entail or depend on distributivity of either kind cannot be maintained. Examples of theories of this defective kind will be discussed. Finally, we will describe a theory of conjunction which does not entail or depend on either syntactic or semantic distributivity and is nevertheless able to support a broad range of properties of conjunction, particularly `non-constituent conjunction' and conjunction of unlike categories.