564 Lecture 22 Nov. 9, 1999

1 Homework

1. Lambda-notation simplifications (#5 on last time's handout):

(a) [λx D_e . x is a cat](Felix)
= 1 iff Felix is a cat.

(b) [λx D_e . x is calico]([[**Felix**]])
= 1 iff [[**Felix**]] is calico. (by λ)
(= 1 iff Felix is calico) (by L.T.)

(c) [λx D_e . [λy D_e . y chased x]](Rover)(Felix)
= 1 iff Felix chased Rover.

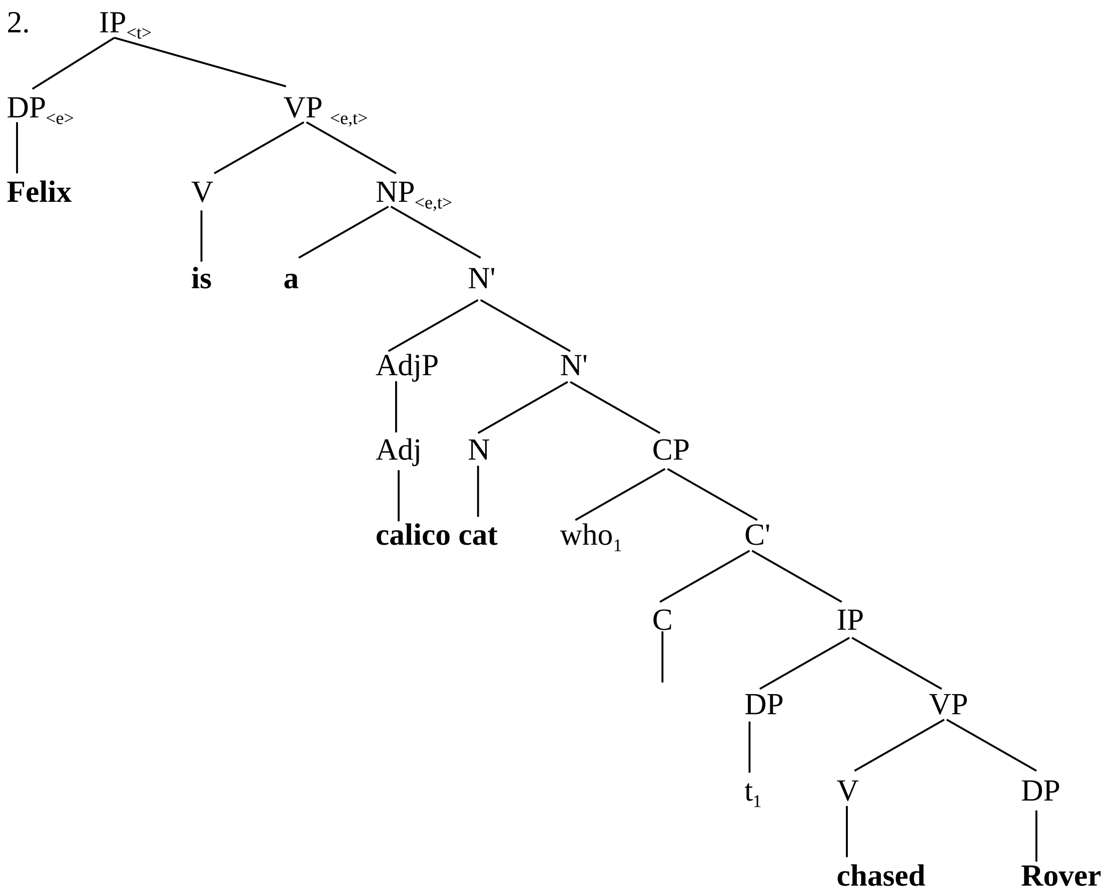
(d) [λx D_e . [λy D_e . y chased x](Rover)](Felix) =
=1 iff Rover chased Felix.

(e) [λx D_e . [[**calico**]](x)=[[**cat**]](x)=1]
= [λx D_e . [λy D_e . y is calico](x)=[λz D_e . z is a cat](x)=1] (by L.T.)
= [λx D_e . x is calico and x is a cat] (by def. of λ)

(f) [λf $D_{<e,t>}$ and there is only one x D_e s.t. f(x) =1 . the unique y D_e s.t. f(y)=1]([λx D_e . [[**calico**]](x)=[[**cat**]](x)=1]) =
= the unique y D_e s.t. [λx D_e . [[**calico**]](x)=[[**cat**]](x)=1](y)=1 (by def. of λ)
= the unique y D_e s.t. y is calico and y is a cat (by def of λ)
= the unique calico cat D_e.

(g) [λx D_e . [λw D_e . [λy D_e . y is a cat](w)=[λz D_e . z chased Rover](w)=1](x)=[λv D_e . v is calico](x)=1](Felix)
=[λw D_e . [λy D_e . y is a cat](w)=[λz D_e . z chased Rover](w)=1](Felix)=[λv D_e . v is calico](Felix)=1 (by def of λ)
= [λy D_e . y is a cat](Felix)=[λz D_e . z chased Rover](Felix)=1=[λv D_e . v is calico](Felix)=1 (by def of λ)
= 1 iff Felix is a cat and Felix chased Rover and Felix is calico
(by def of λ)
= 1 iff Felix is a calico cat who chased Rover (simplfying to equivalent metaEnglish).

2.



(a) [[IP]]

=[[VP]]([[DP]]) by F.A. applied to IP

=[[VP]]([[**Felix**]]) by N.N. applied to DP.

=[[N']]([[**Felix**]]) by N.N. applied to VP 2x

=[λx D_e **.** [[AdjP]](x)=[[N']](x)=1]([[**Felix**]]) by P.M. applied to N'

=[λx D_e **.** [[**calico**]](x)=[[N']](x)=1]([[**Felix**]]) by N.N. applied to AdjP2x

=[λx D_e **.** [[**calico**]](x)=[λy D_e **.** [[N]](y)=[[CP]](y)=1](x)=1]([[**Felix**]])
by P.M. applied to N'

= [λx D_e **.** [[**calico**]](x)=[λy D_e **.** [[**cat**]](y)=[[CP]](y)=1](x)=1]([[**Felix**]])
by N.N. applied to N

= [λx D_e **.** [[**calico**]](x)=[[**cat**]](x)=[[CP]](x)=1=1]([[**Felix**]])
by def of λ applied to λy

= [λx D_e **.** [[**calico**]](x)=[[**cat**]](x)=[λy D_e **.** $[[C']]^{1/y}$](x)=1]([[**Felix**]])
by P.A. applied to CP
(and removing one of those superfluous "=1" in the condition on λx)

$= [\lambda x \quad D_e \,\mathbf{.}\, [[\mathbf{calico}]](x)=[[\mathbf{cat}]](x)=[\lambda y \quad D_e \,\mathbf{.}\, [[IP]]^{1/y}](x)=1]([[\mathbf{Felix}]])$

by N.N. applied to C'

$= [\lambda x \quad D_e \,\mathbf{.}\, [[\mathbf{calico}]](x)=[[\mathbf{cat}]](x)=[\lambda y \quad D_e \,\mathbf{.}\, [[VP]]^{1/y}([[DP]]^{1/y})](x)=1]([[\mathbf{Felix}]])$

by F.A. applied to IP.

$= [\lambda x \quad D_e \,\mathbf{.}\, [[\mathbf{calico}]](x)=[[\mathbf{cat}]](x)=[\lambda y \quad D_e \,\mathbf{.}\, [[VP]]^{1/y}([[\mathbf{t_1}]]^{1/y})](x)=1]([[\mathbf{Felix}]])$

by N.N. applied to DP.

$= [\lambda x \quad D_e \,\mathbf{.}\, [[\mathbf{calico}]](x)=[[\mathbf{cat}]](x)=[\lambda y \quad D_e\mathbf{.}[[V]]^{1/y}([[DP]]^{1/y})([[\mathbf{t_1}]]^{1/y})](x)=1]([[\mathbf{Felix}]])$

by F.A. applied to VP

$= [\lambda x \quad D_e \,\mathbf{.}\, [[\mathbf{calico}]](x)=[[\mathbf{cat}]](x)=[\lambda y \quad D_e\mathbf{.}[[\mathbf{chased}]]^{1/y}([[DP]]^{1/y})([[\mathbf{t_1}]]^{1/y})](x)=1]([[\mathbf{Felix}]])$

by N.N. applied to V

$= [\lambda x \quad D_e \,\mathbf{.}[[\mathbf{calico}]](x)=[[\mathbf{cat}]](x)=[\lambda y \quad D_e\mathbf{.}[[\mathbf{chased}]]^{1/y}([[\mathbf{Rover}]]^{1/y})([[\mathbf{t_1}]]^{1/y})](x)=1]([[\mathbf{Felix}]])$

by N.N. applied to DP

(if you want to get kind of tricky, this next step is possible: substituting "x" for "y" by def of lambda - *in the modification to the assignment function*:

$= [\lambda x \quad D_e \,\mathbf{.}\, [[\mathbf{calico}]](x)=[[\mathbf{cat}]](x)=[[\mathbf{chased}]]^{1/x}([[\mathbf{Rover}]]^{1/x})([[\mathbf{t_1}]]^{1/x})=1]([[\mathbf{Felix}]])$

by N.N. applied to DP

and now, being even more tricky, substitute [[**Felix**]] for x everywhere you see it, by the def of λ — including inside the assignment functions:

$= [[\mathbf{calico}]]([[\mathbf{Felix}]])=[[\mathbf{cat}]]([[\mathbf{Felix}]])=[[\mathbf{chased}]]^{1/[[\mathbf{Felix}]]}([[\mathbf{Rover}]]^{1/[[\mathbf{Felix}]]})([[\mathbf{t_1}]]^{1/[[\mathbf{Felix}]]})=1]$

and this is as far as you can get without applying either Lexical Terminals or Pronouns and Traces. Now I'll apply Pronouns and Traces:

$= [[\mathbf{calico}]]([[\mathbf{Felix}]])=[[\mathbf{cat}]]([[\mathbf{Felix}]])=[[\mathbf{chased}]]^{1/[[\mathbf{Felix}]]}([[\mathbf{Rover}]]^{1/[[\mathbf{Felix}]]})([[\mathbf{Felix}]])=1]$

by P.T. applied to $[[\mathbf{t_1}]]^{1/[[\mathbf{Felix}]]}$

And this is really as far as you can get without applying L.T.

3.

$= [[\mathbf{calico}]](Felix)=[[\mathbf{cat}]](Felix)=[[\mathbf{chased}]]^{1/Felix}([[\mathbf{Rover}]]^{1/Felix})(Felix)=1]$

by L.T. applied to [[**Felix**]]

$=[\lambda x \quad D_e \,\mathbf{.}\, x \text{ is calico}](Felix)=[[\mathbf{cat}]](Felix)=[[\mathbf{chased}]]^{1/Felix}([[\mathbf{Rover}]]^{1/Felix})(Felix)=1]$

by L.T. applied to [[**calico**]]

$=1$ iff Felix is calico and $[[\mathbf{cat}]](Felix)=[[\mathbf{chased}]]^{1/Felix}([[\mathbf{Rover}]]^{1/Felix})(Felix)=1$

by def. of λ applied to λx

=1 iff Felix is calico and [λx   D . x is a cat](Felix)=[[**chased**]]$^{1/Felix}$([[**Rover**]]$^{1/Felix}$))(Felix)=1
by L.T. applied to [[**cat**]]

= 1 iff Felix is calico and Felix is a cat and [[**chased**]]$^{1/Felix}$([[**Rover**]]$^{1/Felix}$)(Felix)=1
by def. of λ applied to λx

= 1 iff Felix is calico and Felix is a cat and [[**chased**]]$^{1/Felix}$(Rover)(Felix)=1
by L.T. applied to [[**Rover**]]$^{1/Felix}$

=1 iff Felix is calico and Felix is a cat and [λx   D [λy   D**.** y chased x]](Rover)(Felix)=1
by L.T. applied to [[**chased**]]$^{1/Felix}$

= 1 iff Felix is calico and Felix is a cat and [λy   D**.** y chased Rover](Felix)=1
by def. of λ applied to λx

= 1 iff iff Felix is calico and Felix is a cat and Felix chased Rover
by def. of λ applied to λy

SO: the meaning of the sentence "Felix is a calico cat who chased Rover" is true iff Felix is a calico cat who chased Rover.

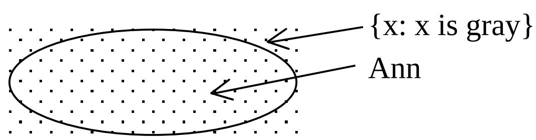
2 Generalized Quantifiers

Last time we saw that quantified DPs cannot be interpreted as individuals in D, nor as subsets of D. How can they be interpreted?

In our terms, there's two possible ways to approach the problem: one is to assign them some type that's not <e> that doesn't take the VP as an argument, and make up a new rule of interpretation to combine the meanings of the VP and the DP, or, more parsimoniously, we can assign quantified DPs some type that takes the VP as an argument and make them combine with it via functional application. In fact, this latter approach is the one H&K adopt, and we'll see why; it makes good sense in terms of our treatment of verbs as relations between individuals and truth-values. We're going to see that quantifiers are a relation between *sets* and truth values.

1. Set-theoretic picture of intransitive predicates, e.g. *Ann arrived, is gray, is a cat*:

(a) If *Ann is gray* is true, then this is a good representation of the world:

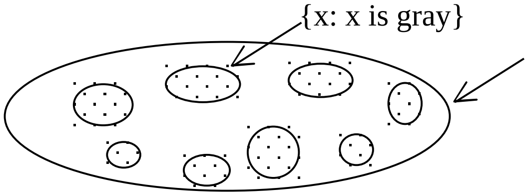


(b) Our semantics says "Ann is gray" is true iff Ann   {y: y is gray}: it says that the meaning of **gray** is a function (relation) between individuals and truth values, conditioned by the grayness of the individual

2. Set-theoretic picture of intransitive quantifiers, e.g. *Everything arrived, is gray, is a cat:*

(a) If *Everything is gray* is true, then this is a good representation of the world:

{x: x is gray}



{X : D X} (or, in words: the set of sets X such that D is a subset of X.)

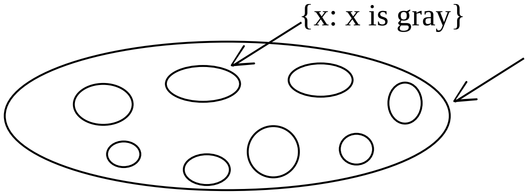
(b) Our semantics is going to say that "Everything is gray" is true iff every individual in D is also in {x: x is gray}, that is, if D {x: x is gray}. That is, our meaning for **everything** is a function between sets and truth values, conditioned by the all-encompassing nature of the set – "Everything is gray" will be true iff [λx D **.** x is gray](x) =1 for all x D,.

2. (a) [[**everything**]] = [λf $D_{<e,t>}$ **.** for all x D_e , f(x) =1]

(b) [[**nothing**]] = [λf $D_{<e,t>}$ **.** for no x D_e , f(x) =1]

"Nothing is gray"

{x: x is gray}

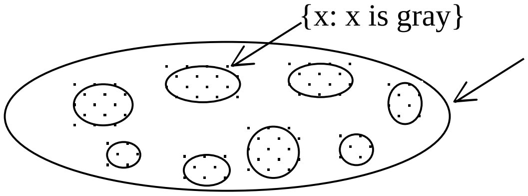


{X : D ∩ X = } (in words: the set of sets X such that no member of D is in X)

(c) [[**something**]] = [λf $D_{<e,t>}$ **.** for some x D_e , f(x) =1]

"Something is gray"

{x: x is gray}

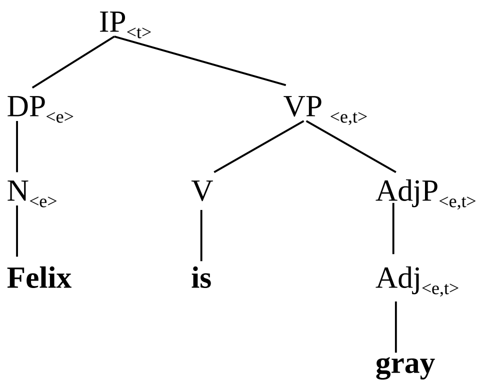


{X : D ∩ X ≠ } (in words: the set of sets X such that at least one member D is in X.)

So, here's where we're suddenly beyond a first-order logic: moving into talking about relations between sets and truth values (second-order), rather than relationships between individuals and truth values. (What kinds of relations do you think "third-order" logic would be able to deal with?)

3. Let's calculate the truth-conditions for two sentences, "Felix is gray" and the sentence "Something is gray" given the denotations we've invented above:

(a)



(a’) [[IP]]

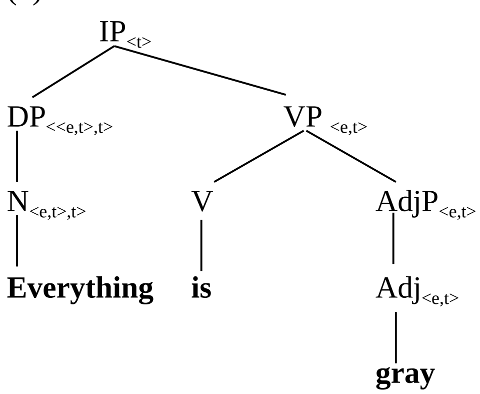
= [[VP]]([[DP]]) by F.A.

= [[**gray**]]([[**Felix**]]) by N.N. 4x

= [[λx D_e **.** x is gray]](Felix) by L.T. 2x

=1 iff Felix is gray.

(b)



(b’) [[IP]]

=[[DP]]([[VP]]) (by F.A)

=[[**Everything**]]([[**gray**]]) (by N.N 4x)

= [λf $D_{<e,t>}$ **.** for all x D_e,, f(x)=1]([λy D_e **.** y is gray]) (by L.T. twice)

= 1 iff for all x D_e,, [λy D_e **.** y is gray](x)=1] (by def. of λ applied to λf)

= 1 iff for all x D_e,, x is gray (by def of λ applied to λy)

3 Avoiding the problems of lecture 20:

Let's just consider one case in detail: the case of the excluded middle. What happens if we say "Everything is concrete or everything is abstract", where {x D_e **.** x is concrete} {y D_e **.** y is abstract} = D_e . We've seen, earlier, that for such predicates, a statement like "Ann is concrete or Ann is abstract" has to be a tautology — if an element of D is not one, then it's certainly the other, because together "concrete" and "abstract" exhaust D. How does our new definition for "everything" avoid this problem? Because "everything" says something about function it takes as an argument, it's the case that what it says about "concrete" has nothing to do with what it says about "abstract": it's a function that maps characteristic functions of sets to truth values depending on the particular set's relationship to D, and it is entirely possible that neither "concrete" nor "abstract" bear the particular relationship to D that "everything" specifies.

4. *Case of the excluded middle*

(a) {x : x is concrete} {y : y is abstract} = D_e

(b) "Ann is concrete or Ann is abstract" has to be a tautology

(c) Why isn't "Everything is concrete or everything is abstract" a tautology?

It is entirely possible that neither "concrete" nor "abstract" bear the particular relationship to D that "everything" specifies.

5. *Case of superset-subset relationship:*

(a) {x : x came yesterday morning} {x : x came yesterday}

(b) If "Ann came yesterday morning" is true, then "Ann came yesterday" must be true.

(c) Why is it the case that "Nothing came yesterday morning is true" doesn't entail "Nothing came yesterday"? Similarly, it is entirely possible that "came yesterday" and "came yesterday morning" simply differ w/r to the particular relationship to D that "nothing" specifies; there's no necessary connection between them.

Similarly for all our other cases. "Nothing" is also a function of type <<e,t>,t>. In that case, "Nothing came yesterday morning" and "Nothing came yesterday" don't have to have anything to do with each other. As long as [[**came yesterday**]] and [[**came yesterday morning**]] aren't the same function (which they're not), then one can belong to the set of sets denoted by "Nothing", and the other can not belong to the set of sets denoted by "Nothing" – there's no requirement that they relate to each other. (Consider Ann and Marie, and let's say they bear some special relationship to each other, like Ann is Marie's mother. There's nothing about that relationship that entails that "Ann is a doctor" and "Marie is a doctor" both have to be true. This line of argument works the same way: just because [[**came yesterday**]] and [[**came yesterday**

morning]] bear a special relationship to each other, that doesn’t mean that “Nothing” has to be true of both functions.

The same argument goes through for all our other cases of failed entailments. We won’t, for the moment, consider how we capture the cases of syntactic reorganization; H&K deal with it on pages 143-145, but we’ll see it in greater detail later.

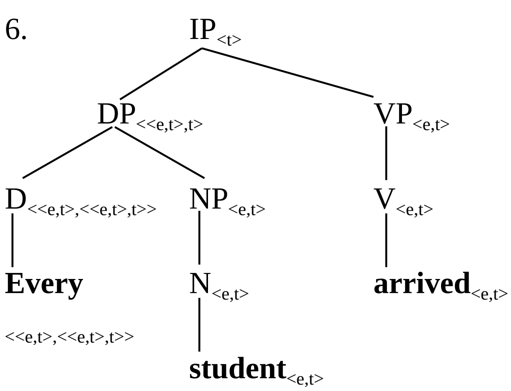
4 2-place quantifiers

Above, we’ve treated “everything”, “something” and “nothing” as monomorphemic words, which just take one argument, a function of type <e,t>. However, it’s obvious that there have to be quantifiers which take more arguments than that. Consider the following:

5. (a) Every student arrived.
   (b) No student arrived.
   (c) Some student arrived.

In these cases, rather than describing the set of arrived things as bearing a particular relationship to D, rather, the set of arrived things bears a particular relationship to the set of students. Basically, the difference between the quantifier here and the quantifier "everything" we treated above is that the latter was intransitive (taking only one predicate as its argument) and this one, the former, is transitive, expressing a relationship between two predicates. In these cases, the quantifier *every*, *no*, or *some* (or any of the other more complex ones we saw earlier) is in the D position, and its sister is an NP (here, "student"), which it takes as its argument; then the whole quantified DP takes the VP as its argument. Assuming all combination is still by functional application, we can represent the sentence in 5(a) as 6 below, with the nodes bearing the types indicated.

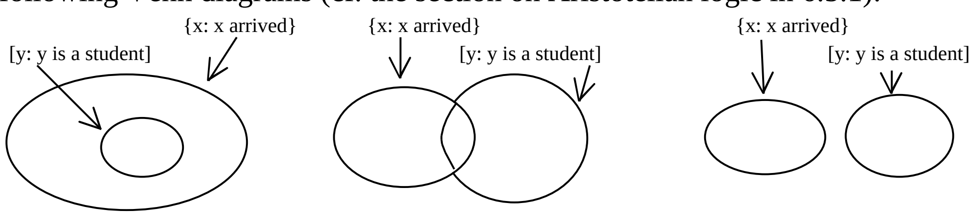
6.



How can we characterize the relationship between the <e,t> functions which is represented by **every**? In the case of verbs, we just use the verb as a descriptor of its own meaning. Above, we characterized the meaning of **everything** as being conditioned by the relationship of the VP set to D_e. Here, we're simply going to characterize the meaning of **every** as being conditioned, not by the relationship of the VP set to D_e, but by the relationship of the VP set to the set described by the NP. Here's the denotation of the three determiners we've got under consideration:

7. [[**every**]] $= [\lambda f \quad D_{<e,t>} \,.\, [\lambda g \quad D_{<e,t>} \,.\, \{x: f(x)=1\} \quad \{y: g(y)=1\}]]$
   [[**some**]] $= [\lambda f \quad D_{<e,t>} \,.\, [\lambda g \quad D_{<e,t>} \,.\, \{x: f(x)=1\} \cap \{y: g(y)=1\} \neq \quad]]$
   [[**no**]] $= [\lambda f \quad D_{<e,t>} \,.\, [\lambda g \quad D_{<e,t>} \,.\, \{x: f(x)=1\} \cap \{y: g(y)=1\} = \quad]]$

This says, "every" takes a function as its argument (in this case "student) and produces a function that takes another function as its argument (in this case, "arrived") and spits out the truth value "1" iff the char. set of the first function is a subset of the char. set of the second function – i.e. iff the set of students is a subset of the set of arrived things. "Some" takes two functions as arguments and spits out "1" iff the intersection of the first set and the second set is not empty, i.e. iff the intersection of the set of students and the set of arrived things is not empty – i.e. iff there are some students who arrived. Finally, "no" takes two functions as arguments and spits out "1" iff the intersection of the first set and the second set *is* empty, i.e. if there are no students who arrived. Set theoretically, we can represent the relationships expressed here in the following Venn diagrams (cf. the section on Aristotelian logic in 6.5.1):



every **some** **no**

I'll spare you the exercises on the mathematical equivalence of 2-place relations between sets, functions from pairs of sets to truth values, functions from sets to sets to truth values (what we're using), and functions from sets to sets of sets (what H&K ask for in the exercise on p. 150). Just keep in mind that in general it works just like our denotations for transitive verbs worked, w/r to a set of pairs of individuals, functions from pairs to truth values, Schönfinkelization of those functions, etc, only substituting sets for individuals. And when you're having fun this weekend, think of me kindly.

5 Homework

Don’t think you’re getting away without any homework, however. Read (up to) H&K section 6.6 on formal properties of relational determiner meanings, and do the exercise on p. 152 finding out which determiners have each property and which don’t. A sample list of test determiners is:

At most one
Exactly three
Few
No
More than two
Some
Three
All
Most

Read “<X,Y> R_δ” as “δX are/is Y”, where δ is the quantifier you’re trying to test. And X   Y is equivalent to saying “All X are Y”.

(*A Thought*: This *feels* a lot more complicated than finding the properties of regular predicates (is “love” reflexive? is “sibling-of” reflexive?) but in principle it’s exactly the same, only you’re doing it for determiners, and there’s a few more complicated relations available to test because determiners relate sets, not just individuals. Why are we doing it? Because, just as natural language has mechanisms that are sensitive to set-theoretic properties like reflexivity ("John loves himself" not "John loves him" for regular predicates, it has mechanisms that are sensitive to the set-theoretic properties of quantifiers.)

Organize your answers something like this:
PROPERTY α:
determiner 1: yes/no on property α
"an example sentence you tested"
determiner 2:
...
PROPERTY β:
determiner 1 yes/no on property β
"an example sentence you tested"
etc..

When you've tested these determiners, organize them into groups according to which tests they pass and which ones they fail. Don't worry if it seems hard to reason about some of them; just write "can't seem to tell" next to those.

For your test predicates A B and C, use “students”, “smokers” and “women”.

(i) *Example* : Finding out if “all” is reflexive:
Definition of Reflexive given by H&K: <A,A> R_{δ} .
So the test sentence for "all" will be: Is “All A are A.” true for every predicate A?
If it’s clear that this must be true for all predicates, then “all” is reflexive. (Are all students students? Are all women women? seems clear that “all” is reflexive.
Now, try “At most one”: At most one woman is a woman. At most one student is a student. seems clear that “at most one *isn’t* reflexive)... and so on down the list. (Are any of them *antireflexive* -- i.e. is the test sentence *false* for every predicateA?)

(ii) *Example*: to find out if “most” is left upward monotone, do the following:
If all A are B and most A are C, then are most B C?
If all students are smokers, and most students are women, then does it follow that most smokers are women? (no, doesn’t seem to).
Now try with “few”: If all students are smokers, and few students are women, then does it follow that few smokers are women? (no, doesn’t seem to).
How about “more than two”? If all students are smokers, and more than two students are women, then does it follow that more than two smokers are women? (Yes! aha! “more than two” is left upward monotone).
(they call it this because the leftmost set (B) in the conclusion of the argument is a superset of A – so the fact that if δA are C, then δB are C means that δ is licensing an inference from the subset A to the superset B, hence *upward*, when A and B are the leftmost arguments in the argument (C being the rightmost)).

(iii) *Example*: to find out if “some” is right downward monotone:
if all A are B and some C are B, then some C are A.
if all women are students and some smokers are students, then some smokers are women. (nope).
Try “no”: if all women are students and no smokers are students, then no smokers are women (yep – “no” is right downward monotone).
(They call it that because the rightmost set in the last clause of the argument (A) is a subset of B – so if δC are B, then δC are A is a superset-to-subset inference (hence *downward*) affecting the rightmost member of the pair).

***Note**:* get their questions from their homeworks from last time over email, or at least next class.