Variable Free Semantics (and Direct Compositionality)
UCLA Minicourse
May 21-25

Pauline Jacobson
Dept. of (cognitive and) Linguistic Science(s)
Brown University
pauline_jacobson@brown.edu

I. Goals

A. Program of direct compositionality (cf., Montague, 1973)

- syntax works "bottom up" to specify well-formedness of larger expressions in terms of well-formedness of smaller ones
- compositional semantics works in tandem: to supply interpretation of larger expressions in terms of meanings of smaller ones
- moreover: "meaning" = model-theoretic interpretation, not a representation (so semantics directly assigns a model-theoretic interpretation to each expression as it is "built" in the syntax)

Consequences:

- no intermediate level of LF (which is itself assigned a model-theoretic interpretation)
- no extra set of rules mapping surface representations into LFs (or vice-versa)
- no need for the syntax to work - in part - "bottom up" and for the semantics to then "go back" and compute the meaning of the LF representation "bottom up"

B. Relationship between Direct Compositionality and Variable-Free Semantics

- various arguments for LF are based on an (arguably) mistaken notion about binding
- binding is not a relationship between two actual NPs/DPs/traces/variables
- hence, many arguments requiring positing of extra structure/hidden pronouns/hidden variables, traces disappear

NOTE: VF Semantics does not depend on Direct Compositionality

C. Simplification of model-theoretic apparatus

- no need for "assignment functions" or variables as part of the semantic machinery
- all meanings are good, healthy model-theoretic objects

D Unify a bunch of disparate looking phenomena

- paycheck pronouns, functional questions, unexpected connectivity effects in copular sentences, functional readings of other expressions (Mitchell/Partee expressions), unexpected inferences, ATB Binding:

---> all instances of the same phenomenon

- NOTE: This result is (in part) translatable into the standard theory. Thus this holds even if variable-free turns out to be incorrect.

E. The analyses of a variety of phenomena are simplified in the variable-free program

• the relevant phenomena arise directly from the mechanisms needed for pronominal binding in general

the phenomena which come for free and/or (arguably) have simpler analyses than in the standard approach:

functional questions; answers to functional questions; unexpected inferences; unexpected connectivity effects in copular sentences; Across-the-Board binding cases; impossible ATB binding cases; i-within-i effects; some apparent exceptions to i-within-i effects; some apparent exceptions to WCO: paycheck pronouns; Mitchell-Partee expressions and other NPs with functional readings; gender of paycheck pronoun; Pied-Piping; interaction of ACD and Pied-Piping

phenomena which are no harder:

Weak Crossover, interaction of Weak Crossover with functional questions, interaction of WCO and i-within-i with Mitchell/Partee expressions and especially with paycheck pronouns in Bach-Peters sentences (cf., Jacobson, 1977); ACD and some interactions with bound pronouns

F. But, at what cost?

Claim: variable-free machinery is no more complex than standard view
(standard complexities often hidden by not being spelled out)

II. <u>Program of Direct Compositionality and a little bit of Categorial Grammar apparatus</u>

A. Basic idea of direct compositionality

linguistic expression: <[phonological form]; syntactic category; meaning>
Meaning: = model-theoretic object

rules mapping these into other expressions
unary rules; binary rules; others?

Question: what is inside [...]? (i.e., what is the object being built)

Hypothesis 1: (undoubtedly wrong): rules effecting the "phonological" parts of expressions only concatenate expressions
(= cf psgs)
if so, [...] literally can be phonological form; no need for any "structured" representation - just a string

Hypothesis 2: a bit more structure necessary, but not a whole lot
In addition to concatenation, there are "Wrap" rules
(Bach, Dowty, Jacobson, Pollard, Hoeksema & Janda, Morrill, ...)
Wrap rules: treats one expression as an infix (the other as a circumfix)
<u>Additional structure necessary</u>: some way to keep track of the infixation point (edged strings; headed strings (Pollard); divided strings (Joshi); etc.)

Hypothesis 3: lots more structure is necessary; maybe as rich a representation as a tree

within this general program:

- generally,want a small set of very general rules
- usually: unary rules as well as binary rules
  unary rules = "type shift" rules - these map one triple into another - often change meaning and category without changing phonology
- don't want a large set of unary rules (nor of binary or n-ary rules)

One particular version = "Type-Logical" framework
here, will consider something closer to "Combinatory Categorial Grammar"

B. A very brief and rudimentary Categorial Grammar (of, roughly, the "Combinatory Categorial Grammar" variety

Why implement the program in CG?

- probably not necessary, but it gives a transparent way for the syntax to "regulate" the semantic combinatorics
- for this reason, it hooks in very nicely with the general program of direct compositionality (probably why Montague, 1973 adopted it)

Key premises:

(i) syntactic categories: as expressions of distributional facts (i.e., categories encode distributional possibilities)
hence a large part of the syntactic combinatory rules can be "read off" of the categories
(ii) syntactic categories also encode semantic types
(iii) hence semantic combinatorics is predictable from syntactic combinatorics

- atomic categories (possibly primitives are actually features, with "categories" being feature bundles)

- recursive definition of other categories:
  If A is a category and B is a category, then A/B is a category
  intuition: an expression of category A/B will "combine" in some way (by some combinatory rule) with an expression of category B to yield an expression of category A (this is a "non-directional CG; need category-specific syntactic rules to specify actual mode of syntactic combination)

<u>"Directional CG"</u>:

If A is a category and B is a category, then $A/_R B$ is a category

intuition: an expression of category $A/_R B$ will take an expression of category B to its right to yield an expression of category A

If A is a category and B is a category, then $A/_L B$ is a category

If there are wrap rules: will need additional categories:

$A/_W B$ and $A/_I B$ (first says "I'm a wrapper": second says I'm an infix)

• Atomic categories (preliminary): S, NP, N, CP, PP?

Presumably: items listed in underspecified form in the lexicon; rules will add directi
onal features (except in exceptional cases such as postpositions in English - e.g., *ag
o)* and other features (case selection features where case is not lexically specified),
etc.

<u>Syntax/semantics "interface"</u>:

• Semantic type predictable from syntactic category

each syntactic category corresponds to some semantic type
any expression of category A/B is of type $\langle b,a \rangle$, for b the semantic type of e
xpressions of category B, and a the semantic type of expressions of
category A
(ignoring intensions here and throughout)

• Semantic combinatorics predictable from syntactic combinatorics
Example: if an expression of category A/B combines with an expression of
category B, the semantics is functional application

<u>Generalized binary rule schemata</u>

(1) <u>Right Concat</u>:
Let α be an expression: $\langle [\alpha];\ A/_R B;\ \alpha' \rangle$ and β be an expression:
$\langle [\beta];\ B;\ \beta' \rangle$. Then there is an expression γ: $\langle [\alpha\beta];\ A;\ \alpha'(\beta') \rangle$

(2) <u>Left Concat</u>:
Let α be an expression: $\langle [\alpha];\ A/_l B;\ \alpha' \rangle$ and β be an expression:
$\langle [\beta];\ B;\ \beta' \rangle$. Then there is an expression γ: $\langle [\beta\alpha];\ A;\ \alpha'(\beta') \rangle$

(3) <u>Wrap</u>??? - assume each string marked with an "infixation" point |
Let α be an expression: $\langle [x \mid y];\ A/_W B;\ \alpha' \rangle$ and β be an expression:
$\langle [\beta];\ B;\ \beta' \rangle$. Then there is an expression γ: $\langle [x\ [\beta]\ y];\ A;\ \alpha'(\beta') \rangle$
NOTE: this is incomplete, since the phonological result will also have to sp
ecify whose insertion point gets inherited as the insertion point in the new ex
pression (same for all of the other rules, too)

(4) <u>Infixation:</u>
Let α be an expression: $\langle [\alpha];\ A/_I B;\ \alpha' \rangle$ and β be an expression:
$\langle [x \mid y];\ B;\ \beta' \rangle$. Then there is an expression γ: $\langle [x\ [\alpha]\ y];\ A;\ \alpha'(\beta') \rangle$

So: $A/_W B$ says "I'm a circumfix", i.e., I wrap around my argument
$A/_I B$ says "I'm an infix", i.e., I go inside my argument

<u>Adding in some unary rules</u>
Note: want small number of these, and hopefully only "natural" ones

(= "type-shift" rules - except that once embedded in a full CG syntax, any semantic type change will also necessarily be coupled with a syntactic category change)

NOTE: other kinds of unary rules are possible; e.g., ones changing meaning but without changing semantic <u>type</u>; ones just changing phonology; etc.

Example: Generalizing Partee and Rooth's proposals about type-lifting:

<u>Definition</u>: Let f be a function from a to b. Take any x in a. Then $Lift_b(x) = \lambda f[f(x)]$
hence $Lift_b(x)$ is a function (a-->b)-->b

Generalized Lift rule:

(5) <u>$Lift_B$</u>

1. Let α be an expression <[α]; A; α'>. Then there is an expression β:
<[α]; $B/_R(B/_LA)$; $lift_b(\alpha')$>
Abbreviation: $\mathbf{l_b}(\alpha)$ will be used to abbreviate the full triple

2. Let α be an expression <[α]; A; α'>. Then there is an expression β:
<[α]; $B/_L(B/_RA)$; $lift_b(\alpha')$>

NOTE: it becomes an accident that the output preserves word order. This problem goes away under various other conceptions of CG. It can also go away here if one takes syntactic categories to actually be functions from strings to strings. Then the output category is just the lift of the input category, and by definition of lift it follows that it's order preserving.
NOTE: The order-preserving property also becomes automatic in a type-logical implementation

Similar pairs will lift wrappers into infixes and infixes into wrappers

• So, following Partee and Rooth (1983) - assume that ordinary individual-denoting NPs (*Bill, the man,* etc.) are "born" with their lowest type
Bill; NP; b
the; $NP/_RN$; maps a set into its unique contextually salient member
the man; NP; ιx[man'(x)]

• but, these can type lift into their generalized quantifier meanings -
their lifted category will be $S/_R(S/_LNP)$ and their meaning will be of type <<e,t>,t>

• *NOTE:* type-lifting is motivated independently of the program here. Given Montague, ordinary "NPs" like *Bill* and *the man* seem to be able to denote generalized quantifiers. Given Partee and Rooth, one may wish not to list them in the highest meaning but to have a productive lift rule. Once one adds a syntactic side to Partee and Rooth and generalizes this to allow lifting for <u>all</u> categories, one is home free.

<u>One (family of) additional Binary schemata: Function composition</u>

(6) <u>Right composition</u>:

Let α be an expression: <[α]; $A/_RB$; α'> and β be an expression: <[β]; $B/_RC$; β'>. Then there is an expression γ: <[αβ]; $A/_RC$; α' o β'>

<u>Left composition</u>:

Let α be an expression: <[α]; $A/_LB$; α'> and β be an expression: <[β]; $B/_LC$; β'>. Then there is an expression γ: <[βα]; $A/_LC$; α' o β'>

<u>Notes</u>: (1) Steedman proposes two additional cases of "mixed composition" but wh ere these apply only with certain categories; not clear whether these are need ed (depends also on how work this in with Wrap operations)

Again: this can be generalized if categories are taken to be operations on stri ngs

(2) Various ways to fold in function composition with Wrap (see Jacobson, 1992 for one proposal)

NOTE:

- one can recast function composition as a unary (type-shift) rule
- a version of this will actually be done below
- the "unary" version is a "Curried" version of the function composition operator; k nown as the "Geach" rule, **g**, or "Division"
- thus **g** maps a function f into something which then wants to take as argument a f unction h such that it returns the value one would have gotten by function co mposing f with h

  i.e., **g**(f) (h) = f o h

Recasting in this way:

(7)

Let α be an expression <[α]; $A/_RB$; α'>. Then there is an expression δ of t he form: <[α]; $(A/_RC)/(B/_RC)$; λV[λc[α'(V(c))]]>

note the semantics here could equivalently have been written:

λV[α' o V]

So - something which wants a B to its right maps into something which want s an "incomplete B" - a $B/_RC$ to its right - and it "inherits" the information t hat a C is still missing (both in the syntax and in the semantics)

<u>for simplicity:</u> in most of the applications for things like extraction, coordination, etc . I'll directly use function composition instead of the two steps of Geach + a pplication; this is just to keep things less cumbersome

<u>*A quick tour of some previous applicatons*</u>

A. Coordination, including "non-constituent" coordination and Right Node Raising

• Take the generalized (Boolean) *and* of Gazdar (1981), Keenan and Faltz (1985), Partee and Rooth (1983) etc. and add CG syntax here:

(8) "and": <[and]; $(\alpha/_L\alpha)/_R\alpha$; ∏> for α any category whose final result is S

where ∏ = & for two objects of type t, and otherwise

∏ = λf[λg[λx[f(x) ∏ g(x)]]]

note: cross-categorial syntax and semantics of *and* can also be thought of as follows: list ordinary *and* in lexicon as

<[and]; $(S/_LS)/_RS$; &>

and derive all the others by a generalization of the "Geach" rule

Note further: directionality on the slashes are not actually listed in the lexicon, but a re instantiated by general principles of directionality in English:

$X/X \rightarrow X/_LX$ (modifiers go on the left)
$S/X \rightarrow S/_LX$ (subjects of S go on left)
$A/B \rightarrow A/_RB$ (all other functions take arguments

to their right = English as head first)

(9) John ate beans yesterday and peas today. (Dowty, 1989 analysis):

beans: NP; beans' $\rightarrow_{l_{VP}}$ beans; $VP/_L(VP/_RNP)$; $\lambda R[R(beans')]$
("VP" = $S/_LNP$)
yesterday; $VP/_LVP$; yesterday'

combine these by left composition:
beans yesterday; $VP/_L(VP/_RNP)$; yesterday' $\circ$ $\lambda R[R(beans')]$ =
$\lambda R[yesterday'(R(beans'))]$
similarly: peas today; $VP/_L(VP/_RNP)$: $\lambda S[today'(S(peas'))]$

beans yesterday and peas today; $VP/_L(VP/_RNP)$;
$\lambda R[yesterday'(R(beans'))] \sqcap \lambda S[today'(S(peas'))]$ =
$\lambda T[\lambda R[yesterday'(R(beans'))](T) \sqcap \lambda S[today'(S(peas'))](T)]$ =
$\lambda T[yesterday'(T(beans')) \sqcap today'(T(peas'))]$

ate; $VP/_RNP$; ate'

ate beans yeaterday and peas today; VP; $yesterday'(ate'(beans')) \sqcap today'(ate'(peas'))$

Right Node Raising:

(10) Mary loves and John hates model-theoretic semantics

Mary; NP; m $\rightarrow_{l_S}$ Mary; $S/_R(S/_LNP)$: $\lambda P[P(m)]$
loves; $(S/_LNP)/_RNP$; loves'
Mary loves; $S/_RNP$; $\lambda P[P(m)]$ $\circ$ loves' = $\lambda x[\lambda P[P(m)](loves'(x))]$ = $\lambda x[loves'(x)(m)]$
similarly; John hates; $S/_RNP$; $\lambda y[hates'(y)(j)]$

Mary loves and John hates; $S/_RNP$; $\lambda x[loves'(x)(m)] \sqcap \lambda y[hates'(y)(j)]$ =
$\lambda x[loves'(x)(m) \;\&\; hates'(x)(j)]$

model-theoretic semantics; NP; mts'

Mary loves and John hates model-theoretic semantics; loves'(mts')(m) & hates'(mts')(j)

B. Wh - Extraction

- are a number of proposals along these general lines - including
  - original GPSG (Gazdar, Klein, Pullum, and Sag, 1984) proposal
  - Steedman, 1987, 1989

  - various mergings of these two: Jacobson, 1989, Oehrle, 1991, Moortgat, 1988, etc.
- for convenience, will pick Steedman (1987); the others are similar

<u>a (probably oversimplified) account of relative clauses</u>

- let a relative pronoun like *who(m)* be listed as:
  <[who(m)]; (N/$_L$N)/$_R$(S/$_R$NP); λP[λQ[λx[P(x) & Q(x)]]]
  thus it takes a property (set-denoting) argument to its right, and a set denoting argument to its left, and intersects them

  (a) need to generalize to the case of subject relative pronouns
  (b) need to generalize to the case of Pied-Piping, which we will do later
  (c) this puts the entire action into the meaning of the relative pronoun; will suggest l
  ater that this is not correct

(11) every woman who Bill invited

invite; (S/$_L$NP)/$_R$NP; invite'
Bill; NP; b --->$_{\mathbf{l}}$ Bill; S/$_R$(S/$_L$NP); λP[P(b)]

function compose these two:

Bill invited; S/$_R$NP; λx[invite'(x)(b)] (i.e., the set that Bill invited)

who then takes this as argument tot its right, then takes *woman* as argument to its left, and returns the intersection of those two sets
that intersection then occurs as argument of *every*

who Bill invited; N/$_L$N; λQ[λx[λy[invite'(y)(b)](x) & Q(x)]] =
λQ[λx[invite'(x)(b)] & Q(x)]]

woman who Bill invited; N; λx[invite'(x)(b) & woman'(x)]

every woman who Bill invited; S/$_L$(S/$_R$NP); λP[x[(invite'(x)(b) & woman'(x)) --> P(x)]]

C. Interlude: Quantifier Scope Interactions and Quantifiers in object position

- there are a variety of in situ (direct compositionality) ways to handle these
- exactly how best to do this is ultimately important to the viability of the variable-free progr
  am <u>because</u>:
  - standard account = Binders Out approach (Montague 1973 - Quantifying-In; Lak
  off, 1970, McCawlye, 1972 - Quantifier Lowering; May, 1976 - Quantifier Raising)

    - these all posit a level of representation at which an object quantificational N
    Ps may be out of the sentence; where a variable (and/or indexed pronoun) is in its pl
    ace, where this variable is λ-abstracted over, and where the whole thing is then applie
    d to the meaning of the quantifier phrase
    - this used for quantificational things in object position
    - in most accounts this used for quantificational things in subject position (though Montague shows that that is completely unecessary)
    - this used for wide scope quantification
    - <u>and</u> this is used for "variable-binding" in general

• hence an arguable advantage of the standard approach: variable-binding is done by the m echanisms needed for quantifier scope ambiguities

• hence the variable-free program needs to show that no great extra complication ensues

• however, the final story here depends on the precise analysis of quantifier scope ambiguiti es

some in situe approaches:

(a) Hendriks (1988, 1994) which generalizes the "argument position lifting" rules of Partee and Rooth, and gets scopes according to the order in which different argument positions are lifted

(b) Jacobson (1992) - which also generalizes Partee and Rooth, but not quit e as much as does Hendriks, and gets scope ambiguities by a combination of functio n composition and argument position lifts

(c) Steedman (2000)

(d) Barker (2001) - uses "continuations" techniques (a kind of generalized l ifting???)

• issues are complicated by the fact that the data is complex (see, e.g., Szabolcsi, etc. - on th e UCLA work); by the unclarity of how/when/how much wide scope readings out of embedded positions; by choice function alternatives; etc.

D. How above analyses relate to/anticipate Variable-Free semantics

key, axiomatic, non-negotiable assumption made in a lot of work in semantics (an d in syntax):

(i) the only method of semantic combination is by functional application

an even stronger assumption which is often made:

(ii) all functions must be "saturated" - they must get all their arguments for semantic compo sition to proceed

Note that: (1) follows from (ii) but not vice-versa; (i) is compatible with the idea that functi ons themselves can serve as arguments, but (ii) isn't.
Hence, from (ii) follows (i) and a second often axiomatic assumption:

(iii) functions can't serve as arguments
the syntactic correlate:
all subcategorization desires of something must be satisfied (= "Projection Principle" - and hence empty PROs etc.)

Hence:

(12) every woman who Bill invited

invite' - is of tyep <e,<e,t>> - it wants to apply to an individual argument to give a VP-type meaning
hence: the usual assumption:

at some level of representation there is something there in object position of invite whose meaning <u>*supplies*</u> *an individual-like thing*

in particular, there is something there whose meaning supplies a variable

- *where a variable allows us to kind of "temporarily" posit some individual argum ent of* <u>*invite'*</u> *which later can get λ-abstracted over*

two possibilities:

(a) traditional one: posit a level of representation at which <u>who</u> is in object position ; posit that <u>who</u> translates as a variable <u>x</u>; do the semantics off this pre-movement le vle

(b) post 1975 - version: posit a surface trace in this position, which translates as a v ariable <u>x</u> (and do interpretation after movement)

(13) invited t (or, invited who) ---> invite'(x)
Bill invited t (or, Bill invited who) ---> invite'(x)(b)

So, in more detail: claim is that this has an LF which includes as a piece of its representatio n something roughly like:

<u>b invite x</u> or, <u>b invite t</u>

where the model-theoretic meaning assigne to <u>b invite x</u> (or, <u>b invite t)</u> is a function from assignment functions to propositions

in more detail: the model-theoretic interpretation assigned to <u>invite t</u> or <u>invite x</u> is a function from assignment functions to sets

and, the model-theoretic interpretation assigned to <u>t</u> (or, <u>x</u>) is a function from assig nment functions to individuals

where each assignment function is a function from variable names to individuals

<u>NOTE</u> futher claim of standard account of a semantics with variable: <u>every linguisti c expression</u> has as its meaning a function from the set of assignment functi ons to something else

crucial next step: we ultimately want *(who) Bill invited* to denote a property (i.e., a set); i.e. something of type <e,t>

(more accurately, we want it to denote a function from assignment functions to a property. But it is a <u>closed</u> expression - it contains no open variables, so it had better denote a <u>constant function</u> from the set of assignment functions to sets of ind ividuals)

Hence:
LF representation going "higher up the tree" is something like:

<u>λx[b invite x]</u>

- various possiblities here of how/where the λ gets put into the tree: one slogan is " movement creates a λ" - but one needs to spell out just what this means - when/wher e in the derivation do we actually put in the λ?; another possibility is to take the *who* to map into λx at LF

• moreover, just sticking in a λ is not enough, this needs to be given a model-theoretic interpretation

the way the next step proceeds: = the semantics of λ-abstraction (see, e.g., Dowty, Wall and Peters or other texts to see it spelled out)

• essentially this maps an "open proposition" into a "closed property"

• more specifically: b invite x assigns propositions to each assignment function, and it will partiton the set of assignment functions into those which agree on all values except for x; if two assignment functions disagree on the value they assign to x they will be assigned different propositions

• λ-abstracting on x now yields every assignment function the same value, and in particular assigns to each assignment function the set of individuals who b invited

• this set is then intersected with the set denoted by the head

• technically, both *woman* and *who Bill invited* denote functions from assignment functions to sets

• so really the semantic composition is:

$$\lambda g[\text{ woman}'(g) \cap \lambda x[\text{b invited x}](g)]$$

(this is a function from assignment functions to sets which assigns to each assignment function g the intersection of the set assigned by woman' to g and the set that λx[b-invited-x] assigns to g)

NOTE that I am using λ ambiguously and sloppily here, the inner use of λ is as a symbol in the representation language of LF, and the outer use is just for notational convenience to name a model-theoretic object

More standardly:

$$[[\text{woman who b invited t}]]^g = [[\text{woman}]]^g \cap [[\lambda x[\text{ b invited x}]]]^g$$

the moral:

• in the standard theory, we put in a variable to get the semantic composition to involve only functional application

• we later abtract over that variable, so that it no longer serves any purpose

• at this point, we still have variables and assignment functions, but once we λ−abstract over the variable (introduced by the trace) the assignment functions do no work, as we now have a closed expression - which is a constant function from assignment functions to sets

So: **suppose that all expressions were closed in this way - i.e., that everything was a constant function from assignment functions**

• then, the assignment functions do no work

• and can be stripped away

---> Variable Free Semantics

NOTE: The GPSG/CG/Steedman/ etc. account of relative clauses sketched above is variabl e-free

the key: we don't stick a variable in object position to allow functional application to happen and then later λ-abstract over it to create a property

- rather, we simply allow one additional operation: function composition
  (or, perhaps, Geach + application)
- with that, we directly get the meaning of *who Bill invited* to denote a property (i.e., a set) of individuals
- this happens via function composing (type-lifted) *Bill* with *invite'*
- the result of the function composition is:
  λx[invite'(x)(b)]

Hence, this is a variable-free treatment of relative clause formation

the variable-free program: push this line all the way; get rid of assignment functions alt ogether

III. <u>The variable-free program in a nutshell</u>

A. FIRST: SPELLING OUT THE STANDARD THEORY

- as noted above:
  — model-theoretic apparatus includes an additional "layer" - the set of assignment f unctions G
  -- where each assignment function g in G is a function from each variable name to s ome individual

  (there are also variables over other types of things, so this needs to be made more general, but will suffice here)

- "functional application" is not really functional application

(14) Bill walks ---> walks'(Bill)
really is: $[[\text{walks}]]^g$ $([[\text{Bill}]]^g)$
or: λg[walks(g)(Bill(g))]
note: for convenience I am taking <u>Bill</u> here to denote an individual rather than a ge neralized quantifier

- each pronoun comes in the syntax with an index
  - and translates as a variable with the same index
    (for convenience, I'll use he_i as <u>x</u>, and use <u>y</u> etc. as needed)

(15) Every man_i said that he_i won.

meaning of the pronoun: function from assignment functions to individuals
semantic combinatorics up to where binding takes place: as given above in (14)

- one can do a direct compositional approach or an LF approach; they are more or less the s ame
- as to the ultimate "binding", there are two possibilities:
  - "Derived (T)VP Rule" - which goes naturally with direct compostiionality
  - Binders Out - which goes naturally with LF

• though note that all four combinations are possible

(i) Direct Compositional approach:

(16)

won --> won': where this is a symbol for a (constant) function from assignment functions to properties

he_i ---> x (or, really, he_{42} ---> x_{42}); where x is a symbol for a function from assignment functions to individuals

he_i won: as above this maps to a function from assignment functions to propositions, where each assignment function g is assigned the value of won'(g) applied to x(g)

said that he won: similarly; we will abbreviate as said'(won'(x))

how do we do binding:

A type-shift rule: The derived (T)VP rule (Partee, 1972 and many since)

(17) VP' ---> λvar[VP'(var)] (for var a variable over variables)

• may also be coupled with a syntactic bit; see below
• this particular formulation allows for vacuous λ-abstraction
• one could block this by keeping track of the unbound variables within the VP via an indexing system in the syntax, as follows:

Let each node contain a feature IND, whose value is a set of indices
• the IND value of the pronoun he_{42} is x_{42}; etc.
• Feature passing convention in the syntax:

The IND value of a mother is the union of the IND value of each of the daughters

• The derived VP rule revisited:

(18)

Let α be a triple of the form: <[α]; VP [IND: M]; α', for i a member of M>. Then there is an expression β of the form: <[α]; VP [IND: M-i]; $\lambda x_i[\alpha'(x_i)]$>

the informal idea:

said he won • denotes the set of individuals who think x won
• more precisely: a function from assignment functions to sets, assigns to each assignment function a set of individuals, where the particular set depends on what value the relevant assignment function assigns to x

• hence, for a g who assigns a to x, then *said he won* assigns to g the set of individuals who think x won

the Derived VP rule:
• maps this into the set of x's who think that x won

• put differently, it maps this into a constant function from assignment functions to individuals, where each assignment function now picks out the set of thinkers-that-self-won

(19) say'(won'(x)) ---> λx[say'(won'(x))(x)]

• this then taken as argument of the generalized quantifier in subject position

<u>A slight extension of the Derived VP rule</u>:

(20) a. Mary told every man_i that he_i won.
b. Mary showed every man_i to his_i mother

binding from Direct Object position into a more oblique complement:
• adopt the Wrap analysis of, e.g., Bach (1979) (see also Dowty, 1982, Jacobson, 1986, etc. - for an antecedent to some of this, see Chomsky, 1957; for translations of this idea into GB see Larson, 1987, and many since)

(21) *tell | that he won*
show | to his mother where | marks the insertion point
are transitive verb phrases $(S/_L NP)/_W NP$
the "direct object" is Wrapped in

• reformulate the Derived VP rule to be the Derived (T)VP rule
• if the TVP undergoes the rule then the next argument in - i.e., the direct object - will bind

(ii) <u>The Binders Out Approach with LF</u> (Quantifier Lowering: Lakoff, 1971, McCawley, 1972, translated into Quantifier Raising: May, 1976)
and see also Montague (1973) for Quantifying In - technically speaking Montague did not use a level of LF, but he did use a level of abstract syntax with indexed pronouns so it is similar

(22) $[_{QP}$ every man, x] $[_S$ x thinks that x lost]

• key point: is a level at which the binder is pulled out and is replaced by a pronoun and/or indexed variable and/or trace

compositional semantics, step by step:
(23) *thinks that he lost* (or, *thinks that x lost)* - same as above

thinks'(lost'(x)) - where this is a shorthand for a function from assignment functions to sets
x thinks that x lost -
thinks'(lost'(x))(x)
- i.e., a function from assignment functions to propositions

• this they "type-shifts" by the semantics of λ-abstraction
• as discussed above in the case of relative clause composition - this shifts into a <u>closed property:</u> the property of being an x such that x thinks x lost

(this is a function from assignment functions to sets, and assigns each assignment function the set of self-loser-thinkers)
• this then occurs as argument of the generalized quantifier binder

C. A Brief discussion of each of these

- apparent advantage of Binders Out: uses same mechanism for binding as can be used for quantifier scopes; the pulling out of the binder and λ-abstraction can go on with or without a pronoun, and is not tied in crucially to the existence of the pronoun
- BUI: an apparent disadvantage: the positions which sanction wide-scope readings are not the same as those which sanction binding:
  - wide scope objects possible, but binding from object position into subject position is not (= the Weak Crossover generalization)
  - so some extra stipulation needs to be added in to keep out extra readings
- apparent advantage of Derived (T)VP rule: WCO is built in - it follows that binding is to a higher argument slot as this is built into the rule
  - well-known problems: this inherits the problems of a c-command account, includi ng *Every man's mother loves him* - see much literature on this including, most rece ntly, Buring (2001)

- NOTE: Variable-Free approach will be closest to Derived (T)VP approach

- a few other comments:
  - Binding is often taken to be a relationship between actual NPs/DPs/variables/traces in the relevant level of structure
  - This is (sort of) true of the Binders Out approach: binding involves there being t wo variables at LF with the same name
  - Strictly speaking, though, this not quite true of the derived (T)VP approach: this i s a relationship between an argument slot and a variable/trace/pronoun in some synt actic structure

- additional comment: these approaches usually coupled with co-indexation in the syntax
  - hence, there are actually an infinite number of pronouns *he*
  - in Derived (T)VP/direct compositional approach, could probably skip the indices, but still *he* has an infinite number of meanings (it needs to have as its meaning any variable name)

B. Variable-Free Semantics (as done in my papers; other versions are possible and have b een proposed; will return)

Basic ideas:

- no variables in the semantics, no assignment functions (variables for notational co nvenience only)
- all meanings are good, healthy model-theoretic objects
- an expression C which contains a pronoun unbound within C denotes a function f rom individuals to something else (not from assignment functions to something else)
- if two pronouns unbound within C - then its a function from two individuals to wh atever
- pronoun itself then must denote a function from individuals to individuals - in part icular the indentity function on individuals
- binding a relationship between argument "slots"
- no indexing needed in the syntax

(24) Every man_i thinks that he_i lost

he lost --/--> lost'(x)
rather: λx[lost'(x)] (= lost')

(25) a. Every man_i thinks that every $woman_j$ said that [he_i likes her_j].
b. Every man_i thinks that every woman said that [he_i likes Mary].
c. Every man thinks that every woman said that [John likes Mary]

<u>standard story</u>: lowest S is same semantic type in all three cases (a function from assignme nt functions to propositions)
<u>variable-free</u>: lowest S has different semantic type in all three cases:
(a) = <e,<e,t>>
(b) = <e,t>
(c) = t

<u>apparent advantage of the standard story</u>:
- it is true that proposition-like things aren't really propositions, but functions from assignment functions to propositions. That said, everything is a function from assig nment functions to something - the semantic combinatorics is uniform throughout, a nd all S-like looking things have the same kind of meaning.
- But in variable-free, the type changes according to how many unbound pronouns within an expression.
- This looks problematic, since expressions with and without pronouns have the sa me distribution (modulo the distribution of resumptive pronouns).

But, stay tuned......

the meaning of the pronoun:
he' = identity function on individuals = λx[x]
probably: identity function on male individuals; ignore gender for now

combining expressions which contain as-yet unbound pronouns

(26) Every man_i thinks that <u>he_i lost</u>

for now: let: he' function compose with lost'
lost' o λx[x] = λx[lost'(x)] = lost'

(27) Every man_i loves <u>his_i mother</u>

let the item *mother* which occurs with a genitive denote a function of type <e,e>:
λx[the-mother-of' (x)] = the-mother-of'

when this function-composes with the identity function, the result will be the-mother-of'

How do binding

a type-shift rule: ***z*** - which shifts the meaning of a function of type <a,<e,b>>
(to be coupled with a syntactic part below):

(28)

Let h be a function of type <a,<e,b>>. Then ***z***(h) is a function of type <<e,a>,<e,b>>, where ***z***(h) = $\lambda f[\lambda x[h(f(x))(x)]]$ (for f of type <e,a>)

Intuition:

love' is an ordinary relation between two individuals (of type <e,<e,t>>)

z(love') is a relation between individuals and functions f of type <e,e>such that to ***z***(love') some function f is to be an x who loves f(x)

composition of (27):

his mother has the meaning the-mother-of'

this combines with **z**(love') - (NOTE: **z**(love') is expecting as argument a function of type <e,e> - NOT an individual)

loves his mother = **z**(love')(the-mother-of') = $\lambda x[love'(f(x))(x)]$

(i.e., the set of individuals who love their own mother)

this then occurs as argument of the generalized quantifier in subject position

composition of (26):

think' is an ordinary relation between individuals and propositions (of type <t,<e,t>>

z(think') is a relation between individuals and properties P (i.e., type <e,t>) such that to ***z***(think') some property P is to be an x who thinks P(x)

he lost has the meaning lost'

this combines with **z**(think') - (NOTE: again **z**(think') is expecting as argument a function of type <e,t>, - NOT a proposition)

thinks he lost = ***z***(think')(lost') = $\lambda x[think'(lost'(x))(x)]$

(i.e., the set of individuals who think that they lost)

this then occurs as argument of the generalized quantifier in subject position

Revising the combinatorics of expressions containing unbound pronouns, and hooking this into a CG syntax

several problems so far:

(i) The generalization noted above:

Consider any expression C which contains within it some NP. Convert this into an expression C' which has instead an pronoun in the NP position. Then wherever C can occur, C' can also occur.

In other words, expressions with unbound pronouns within them have exactly the same syntactic distribution as corresponding expressions with no unbound pronouns within them. (And this holds regardless of the number of unbound pronouns.)

But crucially the types are different - so we would expect the distribution to be different .

Footnote: the reverse generalization does not seem to be the case: resumptive pronou ns are allowed in certain places where expressions without them are not sanctione d

(ii) How to make sure that the combinatorics work out right?
i.e., what ensures that function composition occurs?

(28) Every man_i thinks that Mary loves him_i

loves-him' = loves'
Mary' = m (assume, listed in its lowest meaning)
so: what's to stop
Mary-loves'him' loves'(m)
(instead of what we want, which is λx[loves'(x)(m)]\

(iii) (bothersome for theory-internal reasons)
the syntactic categories and semantic types don't match

(iv) not clear how to get both nested and crossed binding patterns in cases with more than one pronoun

The solution: everything falls out exactly as expected if function composition is recast as the "Geach" rule followed by application
• this gives a way to hook the semantic type into the syntactic category
• this therefore gives a natural way for the syntax to regulate the semantic combinat orics
• this gives an automatic account of the generalization above
• this breaks composition down into two steps - which gives a kind of intermediate step; the existence of this allows for nexted and crossed bindings
• once this is done, the **z** rule will be combined with a natural syntax, and that in tur n will provide an automatic account of i-within-i effects

<u>Hence</u>: how to regulate the semantic combinatorics up to the point the pronoun is bound:

Introduce a new feature, written with a superscript
If A is a category and B is a category, then A^B is a category
The semantic type of A^B is a function of type <B',A'>
Hence A^B is semantically the same as A/B, but syntactically different:
A/B wants to combine with a B in the syntax; A^B doesn't - this really just records that this expression contains within it an expression of category B which is " unbound" within A

Let any expression which contains (or is) a pronoun which is unbound within that expressio n be an expression of category A^{NP}

Pronoun: in lexicon: NP^{NP}
<[he]; NP^{NP}; $\lambda x[x]$>

Assume that any expression which takes an A^{NP} as argument will give a B^{NP} as result (i.e. , the superscript NP feature is "passed up" - which means that the information that there is an unbound pronoun within the relevant expression is passed up)

To do this: have a category change rule: A/B --> A^{NP}/B^{NP}

Natural semantics: the "Geach" rule

For any function f of type <b,a>, there is a function $\boldsymbol{g}_C$(f) of type <<c,b>,<c,a>>, where **g**(f) = $\lambda V[\lambda w[f(V(w))]]$ (for V of type <c,b> and w of type c)

Example to illustrate the intuition: Take the sentential negation operator ~
We can map this into a VP operator *not* by "Geaching" it:

not' = $\lambda P[\lambda x[\sim P(x)]]$

Another observation: "Geach" is a unary (Curry'ed) version of function composition

Thus [**g**(f)](h) = $f \circ h$

<u>Putting this all together</u>: A new general unary rule:

(29) the **g** rule:
Let α be an expression: <[α]; A/B; α'>. Then there is an expression β:

<[α]; A^C/B^C; $\boldsymbol{g}_c(\alpha')$>

(30) Example: composition of *he lost*

lost; $S/_LNP$; lost' $\text{--->}_{\mathbf{g}}$ lost; $S^{NP}/_LNP^{NP}$; $\lambda f[\lambda x[lost'(f(x))]]$

he; NP^{NP}; $\lambda y[y]$

he lost; S^{NP}; $\lambda f[\lambda x[lost'(f(x))]](\lambda y[y])$ (NOTE: by definition of **g** this will give the same result as if we'd function-composed lost' with $\lambda y[y]$)

= $\lambda x[lost'(\lambda y[y](x))]$ = $\lambda x[lost'(x)]$ = lost'

(31) Composition of *Mary loves him*
(Recall the problem: if *loves him* means <u>loves</u>' - what's to stop this from taking m as argument, which will end up yielding a meaning in which *Mary* is the lovee, not the lover)

loves; $(S/_LNP)/_RNP$; loves' $\text{--->}_{\mathbf{g}}$ loves; $(S/_LNP)^{NP}/_RNP^{NP}$; $\lambda f[\lambda x[loves'(f(x))]]$

him; NP^{NP}; $\lambda y[y]$

loves him; $(S/_LNP)^{NP}$; $\lambda f[\lambda x[loves'(f(x))]](\lambda y[y])$ = $\lambda x[loves'(\lambda y[y](x))]$ = $\lambda x[loves'(x)]$ = loves'

the earlier problem: can this combine with *Mary* in such a way that we just do functional application?

No - the syntax is not right

Mary; NP; m - *loves him* doesn't want any such thing as argument <u>because of its syntactic category</u>
--> **l**
Mary; $S/_R(S/_LNP)$; $\lambda P[P(m)]$

still no way to directly combine this with $(S/_LNP)^{NP}$

--> **g**
Mary; $S^{NP}/_R(S/_LNP)^{NP}$; $\lambda R[\lambda x[\text{lifted-Mary}'(R(x))]] = \lambda R[\lambda x[\lambda P[P(m)](R(x)] = \lambda R[\lambda x[R(x)(m)]]$

Mary loves him; S^{NP}; $\lambda R[\lambda x[R(x)(m)]](\text{loves}') = \lambda x[\text{loves}'(x)(m)]$

Giving the syntax for the z rule:

(32) Unary rule: **z**:
Let α be an expression: <[α]; (A/NP)/B; α'>. Then there is an expression β:
<[α]; $(A/NP)/B^{NP}$; $\boldsymbol{z}(\alpha')$>

Semantics is the same as above; the difference is that this effects syntactic category in the pr edictable way - z'ed expressions want to combine with things of category B^{NP}, not things of category B

NOTE: **z** rule needs to be generalized to handle the case of 3-place verbs in which a subject binds into the lowest argument position

(33) $\mathbf{z}_B(((B/NP)/...A) = ((B/NP)...)/A^{NP}$
Given a function f of type $<a,<d_1, ...,d_n<e,b>>>$, $\boldsymbol{z_b}$(f) is a function of type $<<e,a>,<d_1, ..., d_n<e,b>>>$, where $\mathbf{z}_b(f) = \lambda G[\lambda D_1, ...,D_n[\lambda x[\lambda f[f(G(x))(D_1), ..., (D_n)](x)]]]$
(for G a variable of type <e,a> and $D_1...D_n$ variables of type $d_1... d_n$)

Footnote: There is probably no reason to restict this to NPs here - it would be more genera l to reformulate such that NP here is replaced by a variable. This raises a couple of questions:

(a)
At first glance, it looks like binding only happens to pro<u>nouns</u>, hence the rest riction. However, even if this is true there would be no reason to have this restriction ; it could be that the only lexical items of category A^A are those of category NP^{NP} a nd so it would follow from that that only NPs would partake of the bindintg system.

(b)
However, it has often been suggested that at least some cases of VP Ellipsis i nvolve binding of a kind of VP anaphor (not all can, since it happens across sentenc es, but some may involve binding - cf., Rooth, 1984, Szabolcsi, 1992, Schwarz, 2000, among others). In that case it would actually be beneficial to generalize (33).

<u>Solutions to above problems</u>:

(i) the distributional generalization:

A/B ---> A^{NP}/B^{NP}

so it follows that wherever an expression of category B is allowed, so is an e xpression that looks like a B except that it has an unbound pronoun with it (and the mother category will inherit the information that there is an unbound pronoun within it)

Footnote: recall that some environments call for resumptive pronouns. That's okay too, because they can just be things which subcategorize for A^{NP} complements.
Footnote to the footnote: But the story may not end there. A^{NP} will turn out not to mean "I contain a pronoun" but really to record the semantic type; Mitchell/Partee e xpressions and other things without <u>overt</u> pronouns will be of this category too. But it's unclear whether these really can occur in resumptive-pronoun wishing environm ents; the facts are hazy.

(ii) making sure the combinatorics work out right; this has been shown above

(iii) syntactic category encodes semantic type; no mismatch

(iv) nested and crossed bindings: will return later

• but basically all binding patterns possible, have to do with relative order of applications of **z** and **g**

(34) a. Every man_i thinks that every boy_j said that he_i should walk his_j dog.
b. Every man_i thinks that every boy_j said that he_j should walk his_i dog.

• there can be any number of pronouns bound in any order
• application of **z** followed by application of **g** on *say* will yield one pattern
• reverse yields the other pattern
(see Jacobson, L&P paper for the full details)

<u>Free Pronoujns</u>

(35) He lost; S^{NP}; lost'

i.e., a function from individuals to propositions
• Assumption: listerner computes a proposition (since this how information is conv eyed) by applying this to some contextually salient individual

• Is this less natural than in standard theory? Arguably no.
• Note: standard theory - also not a proposition, but a function from assignment fu nctions to propositions
• Yet propositional information again computed

(Ed Keenan points out: need to get a proposition to compute entailments; <u>all</u> Ss in standard theory are functions into propositions; standard theory generalizes t o the worst case)
• so applied to some "contextually salient"(?) assignment function
• in case of a closed expression, its a constant function, so any assignment function will do; but in the open case some assignment function needs to be picked

III. Prelminary scorekeeping:

- for binding, anyone needs a type-shift rule
  - Derived (T)VP rule or, in Binders Out approach, λ-abstraction
  - **z** is just a different type-shift rule
    - takes place on a much more local domain
    - and makes binding a relationship between argument slots

- **g** is extra here, but:
  - paycheck readings on pronouns will turn out to come for free from the **g** rule
  - some systems need something analogous in the syntax anyway, if use feature-pas sing of, e.g., IND feature
  - Pied-Piping semantics (without reconstruction) will come for free from the **g** rule
- Free pronouns - at least as natural

- model-theoretic apparatus simplified (no assignment functions)

- Will show/claim:
  - a whole bunch of functional phenomena come for free (functional questions, funct ional relative clauses, etc. (in some cases, "almost" for free)
  - can do a bunch of things without reconstruction/LF/extra rules mapping into LF (i.e., with direct compositionality)
  - claim: need to define "alphabetic variants" in order to state, e.g., identity condition s on ellipsis and other such things is an artefact of having variable names; once one moves to variable free and thinks just about meanings this unnecessary (cf., Keenan, 1971)
  - claim: difference in variable names never makes a difference

    BUT: open question, see Heim, 97 which makes crucial use of variable name s

- Some common objections:

(a) it *looks* like natural language makes use of variables, since we have pronoun s which look like variables

Answers:

(i) but there are lots of cases where there are binding effects without overt varia bles/pronouns - one example, Mitchell/Partee expressions:

(36) Every man_i visited a $local_{i/j}$ bar
(can be bound, or remain free)

also: "Better Homes and Gardens expressions" (NPs which easily shift to functional readin gs)

(37) Everyone in Berkeley in the 60's would put eucalyptus leaves on the mantel.
(38) In the 18th century, every landowner buried his grandmother in the garden.

functional questions, functional NPs in unexpected inferences, paycheck pronouns, etc. are all similar cases - where there is binding without an overt bindee

(ii) pronouns don't look like variables: variables crucially are indexed, pronouns are not (one invariant pronoun, modulo gender and case - one meaning = identity function)

(b) Variable-free notation (using combinators) is unreadable, why?

• no explanation for this.

(c) No known case where **z** shows up in the morphology; no lexical **z**

• again, no explanation for this, except to note that variables don't seem to be any more trans parent rendering of the surface syntax

IV. Preview of some of the advantages and also of how this connects up to direct composit ionality

Basic arguments for reconstruction/LF:

- assume that binding is a relationship between two NPs/DPs/traces/variables
- which, moreover, must be in a particular configurational relationship to each other (roughl y, binder must c-command bindee)
- but on the surface there are lots of cases where this doesn't hold; where "small" expressio ns stand alone, yet show binding effects
- i.e., binders without bindees in their c-command domain
- bindees without c-commanding binders

<u>hence</u>: posit a level of representaiton at which things moved around and/or extra copies of t hings made so as to get the right configuraitonal relations

Here: all binding effects very local

- binding - takes place on a very local domain (a relationship between argument <u>slots</u>)
- no need to construct big domains to get binding effects
- effect of the pronoun also very local: it has a different semantic type from an ordinary N P, and it affects the semantic type of every expression that it is in "all the way up" un til it is bound

<u>Typical arguments for LF based on binding</u>:

Binder doesn't seem to outscope (and/or c-command) bindee on the surface

(39) His_i mother, every man_i loves.

- won't do to just say that *every man* has widest scope: gives wrong meaning for (40):

(40) His_i mother, I heard that no man_i loves.

so: usual idea: do interpretation at a level at which *his mother* is in object position
• here: no need to do this
  - is just a functional gap, which follows immediately from the fact that *loves* can u ndergo **z**

z(loves) function composes with *no man*
function composition all the way up, gives:
λf[I heard that no-man z-loves f]

• *his mother* automatically has a functional meaning
and so can be taken as argument of this

Details: one should be able to work these out as an "exercize" (see exercize 1 on the exercize sheet). The only thing to worry about is how to do Topicalization in general. See the exercize sheet for discussion and for a proposal.

• similar remarks for unexpected connectivity effects:

(41) The only woman who no Englishman$_i$ loves is his$_i$ mother. (Geach)

(Similar kinds of observations in Engdahl, 1986 on functional questions)

so: these all cases where the bindee doesn't seem to have an overt binder in the right place; hence posit a level where the bindee is in the right place

ATB Binding: one pronoun - two binders:

(42) Every man$_i$ loves and no man$_j$ wants to marry his$_{i/j}$ mother.

(Dahl, 1981; Jacobson, 1984; Hohle,1991; von Stechow, 1992; Jacobson, 1996, etc.)

--> need reconstruction not only for the scopal effects, but to get two pronouns

this a case where there's also a binder without an overt bindee in the right place

Various cases of binding effects without an overt bindee (i.e., without an overt pronoun) --> posit a pronoun and/or variable in LF

e.g. - Mitchell/Partee expressions:

(43) Every man$_i$ frequented a local$_i$ bar. -->
every man'(λx[x frequented a local-to-x bar])

this a case with a binding effect without an obvious/overt bindee

here, no need to posit hidden bindees

IV. Some Technical Details

A. *Question: How do binding into adjuncts?*

• It might look like there is a problem here. Pronouns are passed up from arguments, by the **g** rule:

$$A/B \text{ ---> } A^{NP}/B^{NP}$$

So how could they be passed up from adjuncts?

Answer: No problem. Since we are allowing free type-lifting, an adjunct can always be an argument. Thus an "adjunct" in CG is nothing more than a function which takes its "modify-ee" as argument. The only difference between this and other functions is that a modifyer is of category X/X. The argument can, however, always "lift over" the function - in this case it lifts over the modifyer - to take that as argument.

Example, in a case not involving binding:

(44) John saw Bill yesterday.

Assume *yesterday* is of category $VP/_LVP$. (NOTE: I'll use VP here as an abbreviation for the category $S/_LNP$.) Then *see Bill* could lift to take this as argument, as follows:

(45) see Bill; VP; see'(b) --->$_{\mathbf{l}}$ see Bill; $VP/_R(VP/_LVP)$; $\lambda f[f(\text{see}'(b))]$

it can now combine with *yesterday* , and the ultimate semantics will be
yesterday'(see'(b))
which is exactly what we would have gotten had we taken *yesterday* as the function and *see Bill* as the argument.

Hence, take:
(46) Every man$_i$ hopes that Mary will dance on his$_i$ birthday.

Details left as an exercize for the ambitious, but the point should be clear.

Two related cases:

(a) Binding directly into the adjunct from the next argument up:

(47) Every man$_i$ danced on his$_i$ birthday.

Details should be doable by the ambitious

(b) "Paul Masson" binding:

(48) Paul Masson will sell no wine$_i$ before its$_i$ time.
(49) Mary fired every man$_i$ before I had a chance to warn him$_i$.

• details here require (i) a way to "Wrap" in the object *every man* and (b) a way to have the combine with the TVP *fire* before this happens (i.e., it can be a TVP modifier as well as a VP modifier). As far as I can see, exactly analogous issues arise in any theory. It is worth pondering exactly what one needs to say in a standard account, and exactly how that same basic idea would be implemented here.

NOTE: There are other ways one could do this too, by introducing some new rules for pass ing up pronoun features (along with the appropriate semantics). But as long as lift is indepe ndently motivated and hence needed anyway, there is no need to do anything new for these c ases.

B. How can one do a case where there are two pronouns bound by the same thing?

Case 1: One pronoun can "bind" the other"

(50) Every man_i thinks that he_i should feed his_i dog.

here *he* can "bind" *his dog* - as shown by sloppy identity with VP Ellipsis:

(51) Every man_i thinks that he_i should feed his_i dog and that Bill should too.
(sloppy reading possible)

This case is no problem (work through as an exercize). Note of course that there's no real s ense in which it's correct to say that *he* binds *his* - binding is a relation between arg ument slots - so its more appropriate to say that *his* is bound by the subject slot o f *feed.*

Case 2: This is the more interesting one. What about cases where one pronoun cannot "bin d" the other.

(52) Every man_i thinks that the woman who loves him_i should love his_i dog.

Thus: we can have n pronouns bound by one "binder" (i.e., one argument slot).

- If no notion of "variable" - how can the two pronouns "correspond" to the same thing?
- Answer: they don't

the woman who loves him should love his dog

ends up denoting a 2-place relation:

$\lambda x[\lambda y[\text{the woman who loves y should love x's dog}]]$ (syntactic category is $(S^{NP})^{NP}$

• they end up being "bound" by the same thing - via two applications of **z** on *think:*

(53) think; (S/NP)/S; think' $\text{--->}_{\mathbf{z}}$ think; $(S/NP)/S^{NP}$; $\lambda P[\lambda x[think'(P(x))(x)]]$

Note: this of type <<e,t>,<e,t>>, so **z** on this result will give something of type <<e,<e,t>>,<e,t>> where the intuition is that the subject position of *think'* will bind both of the newly created argument slots

$\text{-->}_{\mathbf{z}}$ think; $((S/NP)/S^{NP})^{NP}$; $\lambda R[\lambda y[\mathbf{z}(think')(R(y))(y)]] =$
$\lambda R[\lambda y[\lambda P[\lambda x[think'(P(x))(x)]](R(y))(y)]] =$
$\lambda R[\lambda y[\lambda x[think'(R(y)(x))(x)](y)]] =$
$\lambda R[\lambda y[think'(R(y)(y))(y)]]$

this then applied to the two-place relation holding between all x and z such that the woman who loves x should love z's dog

demonstrating the intuition: think' of type <t,<e,t>>:
first shift yields: $<<e_i,t>,<e_i,t>>$
second shift yields: $<<e_i, <e_i,t>>,<e_i,t>>$

Consequence: "merging" of the two pronouns is not until higher/later in the semantic co mpositon: this will have a payoff in that it will give an automatic account of Las nik and Stowell's apparent exceptions to Weak Crossover (see below)

C. What about the case of n pronouns - n binders (in any order of binding)

- just various orders of **g** and **z**!

(54) Every mani thinks that every boyj said that hisj mother loves hisi dog.
(55) Every mani thinks that every boyj said that hisi mother loves hisj dog.

the basic intuition:
his-mother-loves-his-dog' = $\lambda x[\lambda y$[y's mother loves x's dog]]

say; $(S/_LNP)/S$; say' of category <t,<e,t>> --> **z** $(S/NP)/S^{NP}$; $<<e_i,t>,<e_i,t>>$
then **g** on this:

$(S/NP)^{NP}/(S^{NP})^{NP}$; $<<<e_j, <e_i,t>>, <e_j, <e_i, t>>>$

hence: the subject position of *say* binds the innermost argument position of the complement; th e outermost argument position will be "passed up" for higher binding gives (16)

say; $(S/_LNP)/S$; say' of category <t,<e,t>> --> **g** $(S/NP)^{NP}/S^{NP}$; $<<e_i,t>,<e_i, <e,t>>>$
(i.e., this operation creates a new argument position in the complement, and "passes up" the binding of this argument position

$\text{-->}_{\mathbf{z}}$ (the argument-skip variety; as the subject here will be binding)
$(S/NP)^{NP}/(S^{NP})^{NP}$; $<<<e_j, <e_i,t>>, <e_i, <e_j, t>>>$

hence: the subject position of *say* binds outermost argument position of the complement; t he innermost position will be passed up for higher binding (gives (17))

(56) (gives 54): every-man'(***z***(think') (***g***(every-boy'(***g***(***z***(said'))(his-mother-loves'his-dog'))))

(57) (gives 55): every-man'(***z***(think')(***g***(every-boy')($\boldsymbol{z_t}$(***g***(said'))(his-mother-loves-his-dog'))))

<u>Excercize</u>: work through the full derivations. One is done in my *L&P* paper; work through the oth er one.

D. What about Weak Crossover?

• This is built right into the system, in pretty much the same way as the Derived (T)VP rule.

• Assume: <u>only</u> binding rule is **z** - this says "bind to a higher argument slot"

• Comments:

(a) Unlike in a Binders Out approach, nothing extra is needed to get WCO effects; i t is perfectly easy to build right into the system, so there is no "overgenerating" and then adding a constraint to stop that.

(b) However, it doesn't follow from anything deep that things should be this way. It would have been just as easy to formulate instead - or in addition - a backwards bin ding rule ***s*** as follows:

(58) Let h be a function of type <e,<b,a>>. Then **s**(h) is a function of type <e,<<e, b>,a>> where **s**(h) = $\lambda x[\lambda f[h(x)(f(x))]]$

This will allow object position to bind into a pronoun in subect position.

(c) This crucially uses argument order in the combinatorics (reminiscent of c-co mmand) rather than left-to-right order. Could be restated using left-to-right order (s ince there are directional features on the slashes), but is somewhat unnatural.

The literature has gone back and forth on linear order vs. c-command (or, "arg-com mand", "f-command" - take your pick); if linear order turns out to be right it might b e worrisome.

(d) This inherits some of the problems of c-command:

(59) *Every man$_i$'s mother loves him$_i$.*

but see paycheck analysis, most recently Buring (2001)

(e) Lots of other mysteries surrounding WCO; no light shed on these here.

But, note a crucial difference with most other accounts:

Usual view: WCO is a constraint on the relationship between two actual things sitting in so me level of syntactic representation (a pronoun/trace/variable and a binder)

(a) ultimately, aconstraint on rules (Postal, 1971; Jacobson, 1972, 1977; vs. (b) a co nstraint on single level of structure (Wasow, 1972; Chomsky, 1976 and many since etc.)

also, debate about: (a) left-to-right order (Postal, 1971; Cole, 1972; Wasow, 1972; J acobson, 1972, 77; etc.) vs. (b) c-command (Reinhart, 1983 and most things since)

here: constraint on the combinatorics: binding is only between a higher argument p osition and a pronoun (or, "open slot") within a lower argument position

HENCE: in standard view - need to posit hidden stuff at an appropriate level in orde r to get binders and bindees to be in the right configuration to express certai n WCO generalizations; here no need for such extra stuff since the constrai nt constrains the combinatorics of "merging" argument slots

will have crucial consequences for (a) analysis of paycheck pronouns and observatio ns of Jacobson (1977); and (b) analogous analysis of functional questions a nd WCO in Chierchia (1992) (see also Engdahl, 1988 for the same point ab out functional questions)

V. Some Initial Payoffs

A. Functional Questions (Groenendijk and Stokhof, 1983; Engdahl, 1986)

(60) a. Who/Which woman does every Englishman love (the most)?
b. His mother.

- tempting to think of this as a matter of Quantifying-In (i.e., assigning wide scope to) *every Englishman*
  - cf., Karttunen (1977)

Karttunen's semantics for (60), very roughly and informally:

(61) for every Englishman, x: [I ask you] who x loves

- posits a hidden performative, since allows quantifying in only into declaratives
- one could instead try to allow quantification directly into questions

But: are a variety of arguments against quantifying-in approach
see Groenedijk and Stokhof (1983) and especially Engdahl (1986) for detailed argu ments

- one will suffice here:

(62) Who/Which woman does no Englishman love?

semantics is not:

(63) for no Englishman, x [I ask you] who x loves

• above paraphrase relies on Karttunen's hidden performative analysis, but other semantics f
or questions also give wrong reading using wide scope reading for *no Englishman*

solution of Groenendijk and Stofkof (83) and Engdahl (86): (basic semantics of this soluti
on is by now pretty universally accepted)

• is actually a question about a function of type <e,e>

roughly:

(64) what is the function $f_{<e,e>}$ such that
every-Englishman' (λx[x loves f(x)])

• the idea here:
-- this semantics is correct
-- question is how the compositional semantics proceeds so as to get this meaning

how compositional semantics works in standard account
(NOTE: G&S and Engdahl each have slightly different strategies, and neither is exa
ctly like the one given below, but both roughly like this)

first: take a non-functional (ordinary, individual) question:

(65) Who does every Englishman love? The Queen.

• many open questions about:
(i) semantic contribution of *who*
(ii) what/when/where/how supplies the question semantics
(iii) what exactly is the question semantics

• hence, here will simplify and work around these questions:

(66) who_x [does] every man love t_x

t_x ---> x
love t_x ---> love'(x)
every man love t_x every man'(love'(x))
NOTE: in Binders Out approach: first:
every man [t_y loves t_x]
t_y loves t_x ---> love'(x)(y)
---> λ-abstraction over y
λy[love'(x)(y)]
this then is argument of every-man'
which is equivalent to above

in non-Binders Out approach: directly take the meani
ng of the VP as argument of the subject (cf., Montague, 73)

assume that <u>x</u> is then λ-abstracted over:
λx[every-man'(love'(x))]

and this then taken as argument of *who*

NOTE: the details of these last two steps depend in part on the precise analy sis of the meaning of *who*

Comment: • this much like the point made earlier about the standard way to do the semantic s of relative clauses
• first we posit a hidden variable in object position to get the semantics combinatoric s to only involve functional application
• then later we λ-abstract over this variable

Extending the standard analysis to functional questions:

• Assume: trace (or whatever it is that supplies the variable in object position in a normal q uestion) can "translate" either as an individual variable x or as a complex variable f(x)

footnote: Engdahl's implementation involves not traces but "linkes" phrase structure trees, where who supplies the relevant semantics and is in both the gap p osition and the extraction position at surface structure; G&S do things slightly diffe rently; a third implementation given in Chierchia (1991) who posits complex indice s on the traces in the surface syntax

(67) $t_{f(x)}$ ---> f(x)

(for notational convenience, I adopt something like co mplex indices on the trace)

love $t_{f(x)}$ ---> love'(f(x))

• next step: binding of x happens in the way that binding would happen in general:

(a) Binders Out: t_x loves $t_{f(x)}$ ---> loves'(f(x))(x)
then λ-abstract over x ---> λx[loves'(f(x))(x)]
then take this as argument of every-man'
every-man'(λx[loves'(f(x))(x)])

(b) Derived VP Rule:
loves $t_{f(x)}$ ---> loves'(f(x)) --->
loves $t_{f(x)}$ ---> λx[loves'(f(x))(x)]
this then taken as argument of every-man' (which remains in situ):
every-man'(λx[loves'(f(x))(x)])

• next step: recall that in ordinary questions, somewhere along the line the x variabl e which corresponds to the object trace gets λ-abstracted over
• so, in functional questions, the f variable will then get λ-abstracted over by presu mably the same process

---> λf[every-man'(λx[loves'(f(x))(x)])]

this then occurs as argument of the question pronoun *who*

Comment: under this view *who'* (or whatever it is that supplies the question semantics) ha s to be polymorphic

• in ordinary case, it takes as argument (or, as input) something of type <e,e>
• here, it takes as argument (or as input) something of type <<e,e>,t>

the same will be true in my analysis; it would be nice to get this polymorphicity to follow in a more general way (possibly from an independently needed type-shift rule), but it remains to be seen whether this can be done

Further observation of Engdahl:

• functional questions are not just restricted to questions about functions of type <e, e>
• but, can have questions about n-place functions
  (functions of type <e,<e,e>> etc.)

Examples:

• easiest examples involve binding into the *wh* -phrase, which introduces a new problem, but for exposition:

(68) Which poem that he wrote for her did each boy ask each girl to memorize?
The one he wrote for her on Valentine's Day.

(69) a. What did each policeman tell each criminal?
That he would allow him one phone call.
b. What did no policeman tell any of the people he arrested?
That he would allow him one phone call.

Engdahl's solution:

• "gap" can be a variable over n-place functions applied to n-individual variables

(80) Which poem did each boy ask each girl to memorize?
The one he wrote for her on Valentine's Day.

memorize $t_{W(x)(y)}$ ---> memorize'(W(x)(y))
  for W a function of type <e,<e,e>>

Potential problems with some of the details in the standard approach

A. A purely theory-internal problem for a CG analysis (or any other direct compositional analysis without traces):

• have seen: in the ordinary case, can just do function composition for extraction (or, the "Geach" rule)

(81) Who does every Englishman love?
  every Englishman love; every-Englishman' $\circ$ love' =
    λx[every-Englishman'(love'(x))]

• recall: we don't need to posit a variable which later gets λ-abstracted over
• this automatically denotes a function from individuals, because there is a "missing object"

- in other words, extraction "gaps" are just the failure of an expected argument to be introduced

but: in functional questions: this won't extend -

- we need some actual thing (trace, *who* or whatever) to translate as a variable over functions of type <e,e> applied to an individual variable
- in other words, we can't say that an extraction "gap" is just an argument which has not been introduced
- since then there would be no way to get the "missing argument" to have this kind of a complex meaning

B. Non-theory internal problem:

- suppose we are happy to have a trace in object position (which translates as a variable x)
- for functional questions:
  - Why should it also have this additional, complex meaning?

- NOTE: The main point: functional readings are unexpected; we need to posit something extra to get them (allow the trace, or whatever, to have the complex meaning f(x) as well as the simple meaning x)

C. In fact, we see that there are an infinite number of meanings

- Variable over n-place functions applied to n individual variables

Engdahl: let the ordinary (individual) question just be a special case of functional questions in general

this answers B - no extra apparatus for the ordinary reading (it's just a special case)

- in particular: trace (or whatever supplies the meaning) is polymorphic: translates as any variable with n- places applied to n individual variables
- ordinary reading: a special case
  two ways to do this:

(a)

Engdahl's solution: in this case it's just a "0-place function" (a function with no arguments to individuals)
problem: this seems to me to be a terminological trick; a function is something with arguments

(b)

let the individual case just be a matter of a constant function (that is - we still have f(x) as the meaning of the gap, but the answer just supplies a constant function)

Comments: these will work, but it's arguably a "generalizing to the worst case" solution

- functional readings are somewhat unusual, and harder to get than ordinary individual readings
- in strategy here: they come for free in the grammar. On the other hand, they require extra type-shifting, and so arguably are harder to get if we take a "lowest types" strategy as a processing strategy

Functional Questions in Variable-Free Semantics

The ordinary cases:

Standard account:
•need to supply a complex meaning for a gap - and hence some actual item in the gap positi on so as to be the item which can have a complex meaning
• no real reason why the "gap" (trace, or whatever) should have this complex meaning; this u nexpected
crucially, this arises from the assumption that "binding" effect requires an actual s yntactic object - i.e., a variable - in order go get bound
arguably, this is a case of binding, with no overt "bindee"

Claim: (1)
(almost) all the apparatus for this is already present in the binding apparatus used in variable-free
• hence - functional readings for questions are expected
(2)
however, they require more shifting than ordinary readings; hence we are not generalizing to the worst case

Variable-Free solution:

(a)
functional gaps do not have complex meanings; they are just missing functio ns of type <e,e> - no individual argument "variable" needs to be supplied

(b)
it follows immediately that there should be such functional gaps; they are not hing more than the result of the verb having type-shifted by **z**
(or, in some cases, by **g** where there will be binding higher up)

(c)
it follows immediately that there should be gaps over n-place functions; this happens just because there can be any number of shifts by **g** and **z** (i.e., happens in t he same way that multiple pronouns are allowed in general)

(82)
Who does every man love? (using Steedman-style extraction syntax, thoug h this not crucial):

love; $(S/_L NP)/_R NP$; loves' $-->_{\mathbf{z}}$ love; $(S/_L NP)/_R NP^{NP}$; $\lambda f[\lambda x[love'(f(x))(x)]]$
i.e., I'm expecting a pronoun-type of argument to my right (but I won't get it)

every man; $S/_R(S/_L NP)$; every-man'

every man loves; $S/_R NP^{NP}$; every man o **z**(love) =
every man love; $S/_R NP^{NP}$; $\lambda f[every\text{-}man'(\lambda x[love'(f(x))(x)])]$
this then occurs as argument of question pronoun *who*

note: • this is exactly the G&S/Engdahl meaning; it's just that the semantic composition pro ceeds differently

• the fact that the gap can have a functional meaning is an automatic consequence of the fact that *loves* can undergo **z**

Potential complication:

• above, we saw that *who* (or whatever it is which supplies the question meaning) must be polymorphic; it can take as argument an <e,t> and also an <<e,e>,t> (an any more complex Engdahl meanings)
• same thing here - moreover, it must have a generalized syntactic category, such that it can take an $S/_R NP^{NP}$ as argument as well as an $S/_R NP$

• this fact does not come "for free" here (nor does it in the "standard" theory)
• it would ultimately be nice to just list a simple meaning/category for *who* and have the rest derived by a productive type-shift rule

Are we getting this too cheaply?

• Functional readings on questions are difficult and arguably should involve something extra
• Possible answer here: "lowest types" scenario - as a <u>processing strategy</u> (only!) -

• Additional lifts, type-shifts "take more work" - lead to more difficult readings
• Hence, the simplest composition of questions gives the individual reading; functional reading involves **z**
• Note that in "ordinary" pronoun binding case, this not true: it takes no more work to bind a pronoun than to not bind it

• To not bind it requires applying **g** to things to pass up the binding for later; to bind requires applying **z**

Functional questions with a multi-placed function:

(83)

Which poem did every boy$_i$ hope that every girl$_j$ would read? The one he$_i$ composed for her$_j$ for Valentine's day.

• Follows directly from the mechanisms that allow multiple binders and multiple pronouns in general; just combinations of **z** and **g**

details left as an exercise for the ambitious, but basically:

g(**z**(read)) or **z**(**g**(read))

(in the latter case, **z** will have to be the argument skip variety)

footnote question: once we introduce various new argument positions by various applications of **z** and **g** - do we ever get a newly introduced argument position to mistakenly bind?

i.e.: read' of type $<e_1,<e_2,t>>$ $\longrightarrow_g$

$<<e_{3(i)}, e_1>,<e_{4(i)},<e_2,t>>>$

where e_3 and e_4 are "merged" in the semantics

(this notated above by use of (i))

"argument skip" **z** will map this into:

$<<e_{5(j)}, <e_{3(i)}, e_1>>,<e_{4(i)},<e_{2(j)},t>>>$

where we introduce a new argument position, e_5 and merge it with the ultimate subject position, which is e_2

but, one might ask: what's to stop **z** from applying in a different way, where e_4 is what gets merged with the newly introduced argument position e_5?

$<<e_{3(i)}, e_{1}>,<e_{4(i)},<e_2,t>>>$ $\text{--->}_{\mathbf{z}}$
$<<e_{5(i)}, <e_{3(i)}, e_1>>,<e_{4(i)},<e_2,t>>>$

<u>Answer</u>: (i) the syntax of the **z** rule will stop this: things are bound to argument sl ots which are actual syntactic argument slots of the verb; things are not bound to se mantic argument slots which syntactically come out as superscripts

(ii) actually, I'm not even sure that any bad result would happen if one allowe d argument slots introduced by the **g** rule to be merged in (bind) with argument slots via **z**, but I've formulated things here to be conservative

and the rest involves function composition as with extraction in general

- so these are also unsurprising (modulo the polymorphicity of *who)* - what would be sur prising would be if we didn't find these
- on the other hand, they require extra type-shifts, and so could lead to processing difficulti es (this offered very tentatively)

Functional questions with the gap embedded further down

(84)

Which woman does every Englishman think that the Queen should invite?
His mother.

- these follow just the way one gets long distance binding in general

are various possible derivations: one here:

(85) every man o (**z**(think) o **g**(**l**(the queen) o invite)))

use q for the-queen' (in unlifted form)
the queen invite; $S/_R NP$; λx[invite'(x)(q)] --->$_{\mathbf{g}}$
the queen invite; $S^{NP}/_R NP^{NP}$; λf[λy[λx[invite'(x)(q)](f(y))]] =
λf[λy[invite'(f(y))(q)]]
z(think); $(S/_L NP)/S^{NP}$; λP[λx[think'(P(x))(x)]]
z(think) the queen invite; $(S/_L NP)/_R NP^{NP}$;
λP[λx[think'(P(x))(x)]] o λf[λy[invite'(f(y))(q)]] =
λf[λP[λx[think'(P(x))(x)]](λy[invite'(f(y))(q)])] =
λf[λx[think'(λy[invite'(f(y))(q)](x))(x)]] =
λf[λx[think'(invite'(f(x))(q))(x)]]
` (informally: λf[λx[x thinks q will invite f(x)]]
every man; $S/_R(S/_L NP)$; every-man' this composes with above:
every man thinks the queen invite; $S/_R NP^{NP}$;
every-man' o λf[λx[think'(invite'(f(x))(q))(x)]] =
λf[every-man'(λx[x thinks the queen will invite f(x))])]
this then occurs as argument of *which woman*

Functional questions which are functions into propositions:

(86) What does every man believe? That he will win.

- one possibility for the semantic type of CP complements: really fancy individuals - i.e., propositions are a type of individual
  in that case, there is no problem here for anyone; the "gap" following (86) is like an y other gap in a functional question, and is a function of type <e,e>
- if, however, propositions are something different - call them type p - then G&S, Engdahl a ccount needs an extra bit:

  trace after *believe* can translate as a function of type <e,p> applied to an indi vidual variable
- here, nothing extra will ever be needed - whatever kind of gap we have

believe' of type <p,<e,t>> so, by **z** will map into **z**(believe) of type <<e,p>,<e,t>>
and so the "missing argument" here wil of course be of type <e,p>

B. Answers to Functional Questions

- these potentially supply another argument for variable-free, but this depends in part on on e's theory of answers
- hence, will defer this to a slightly different version of the same argument (below)

C. Functional NPs and Unexpected Binding Connectivity in Copular Sentences

(87) The (only) woman who every Englishman loves is his mother. (Geach, 1962)

Once again, this can't be a matter of wide scope:

(89) The (only) woman who no Englishman invited (to his wedding) is his mother

≠ for no Englishman x is it the case that the (only) woman x invited was his mother

analysis in Jacobson (1994) (SALT 4):

- pre-copular NP can have a functional reading
- this comes almost for free from mechanisms developed so far for binding in general

- need two new pieces:

(a) let *the* be polymorphic:

assume ordinary *the*: maps singleton set into unique memeber - hence, of type <<e,t>,e>

allow it to also map a singleton set of functions (of any complexity) into unique member - hence, also of type <<<e,e>,t>,<e,e>>

etc.

NOTE: obvious problems with uniqueness - for anybody's account of *the* - presumably it's looking for a unique contextually salient member

(b) *woman'* is a function of type <e,t> - allow this to type-shift into a set of functions with <u>range</u> woman'

• It would be nice to get this type-shift rule to follow in some more general way; I don't kow how to do that

given this, the rest is automatic:

(i) <u>semantic composition of the pre-copular NP</u> (extends the analysis of functional questions to the case of relative clauses):

(90)

the unique function f which is the intersection of functions with range woman' and the set of functions f such that every-Englishman' **z**-loves f

in more detail:

(91)

every Englishman loves - function composition of every Englishman and **z**(loves), as above:

loves ---> $_{\mathbf{z}}$ loves; $(S/_L NP)/_R NP^{NP}$; $\lambda f[\lambda x[\text{loves}'(f(x))(x)]]$

every Englishman; $S/_R(S/_L NP)$; every-Englishman'

every Englishman $\circ$ loves; $S/_R NP^{NP}$; $\lambda f[\text{every-Englishman}'(\lambda x[x \text{ loves } f(x))])$

let woman' shift as above, to be set of functions g with range woman'

intersect these - exactly as in the normal case of relative clause formation

then take that as argument of *the* - to return the unique such function

footnote: I have suppressed the contribution of *who* -
in most versions of CG, the intersection semantics is supplied by *who*
• if this is the case, this will also have to be polymorphic (in the same way that question pronouns are):
normal *who*: give me two sets of individuals (one at a time) and I'll return the intersection

this *who*: give me two sets of functions (of type <e,e>) and I'll return the intersection

• presumably, whatever mechanism is used to give polymorphicity on the question pronoun will be applicable here as well

• however, later I'll suggest that sticking the intersection semantics into the meaning of *who* is not really right (since that view doesn't generalize to Pied-Piping)

• hence, it works out better to treat *who* as a normal pronoun; and its shiftability into a a pronoun over functions will be automatic (see paycheck pronouns)

Additional comment: need some notion of "natural function" -

(92) The only woman that every Englishman invited to his wedding was his mother.

suppose: each Englishman invited his mother
Charles also invited his aunt
and there were no other invitings

then one can set up a function which pairs Charles with his aunt and all the others with their mothers - yet (92) is still true

in fact, allowing functions to be any kind of pairings gives (92) on the functional analysis the same truth conditions as an analysis in which *every Englishman* is scoped out

• hence, need some notion of what "counts" as a function - see Sharvit, 1996 for discussion of this

(ii) <u>semantic composition of the post-copular NP</u>:

• composition of pre-copular NP is just an extension of the analysis of functional questions - hence it can actually be done just as easily in the standard theory as can functional questions in general

• i.e., no new argument here for the variable-free view; all the extra pieces are the same

• see von Stechow (1991) and Sharvit (1996) for exactly this analysis in terms of the standard theory

• however, the semantics of *his mother* provides an additional argument for the variable-free implementation:

follows from the whole apparatus that *his mother* denotes a function of type <e,e> -
i.e., = λx[the-mother-of'(x)] = the-mother-of function

nothing extra is needed to get this to be the right type!

von Stechow analysis (see also Sharvit, 1996): need an extra step, where we λ-abstract over the open variable in the meaning of *his mother:*

his mother; x's mother ---> λx[x's mother]
shifts "open individual" into closed (constant) function of type <e,e>

Note: there are various analyses on the market of the syntax/semantics of copular Ss, and of the meaning of <u>be</u>; as far as I can tell it doesn't matter what one adopts here
- be can just be identity
- or can take two arguments an X and an <X,t> as in (sort of, Higgins, 1974), Williams (1982), Partee (1982)

  - if the latter approach adopted, need additional shifts: subject shifts from f to <f,t> - but this just part of Partee's general IDENT shift, nothing new needed above and beyond her analysis of ordinary copular Ss

key point: this gives a direct compositional analysis of copular Ss without any kind of reconstruction, and with minimal new apparatus
standard view of binding can do the same thing (see von Stechow's analysis, for example) but with additional work

<u>A note on connectivity effects in copular Ss in general</u>:

- a variety of connectivity effects are found in copular Ss (see especially Higgins, 1974) and so these often taken to necessitate reconstruction anyway

- standard way to think about the binding effect here would be:
  - *his mother* has a bindee, with no c-commanding binder
  - hence, posit a level of LF at which it is "surrounded" by extra stuff to give it a c-commanding binder

(93) The only woman who every Englishman loves is
every Englishman loves his mother

one way to make sense of this LF = Ross (19??, revived in Shlenker, 1998(?)):
take the pre-copular NP to be a concealed question
and the post-copular constituent to be its answer

- since one might want to argue that answers are propositional anyway, not surprising to add in this extra stuff

arguments against this:

(1) complex mapping from surface syntax to LF (no reason to do this if not necessary)
(2)
to use this as a strategy for connectivity effects <u>in general</u>, one needs to say that this is the <u>only</u> representation for copular Ss

- which boils down to saying that <u>be</u> can <u>only</u> take as arguments a question/answer pair
- but why should <u>be</u> be restricted in this way?

(Reason that one needs to have this be the only *be:* this needed to account for cases of ungrammatical Ss showing connectivity effects)

(3) Sharvit (1999):

(94) What John ate and what Mary ate was (altogether) five eggs and the leftover turkey.

(4) Higgins (1974)

(95) His promise was to shave himself.

• Hence, direct compositionality strategy here will have to account for the full range of conn ectivity effects
• ordinary reflexive connectivity is no real problem:

(96) What John is is proud of himself.

Bach and Partee (1981); Szabolcsi (1988) and many others:
• let the meaning of *himself* be entirely "bound up" within the meaning of the AP
• this is just like the treatment here of pronouns in general, but the locality effects o n reflexive will have to be accounted for in some way
• one possibility = Szabolcsi: reflexive itself is an argument reducer

in any case: proud of himself is just λx[proud-of'(x)(x)]]

(97) the property that John has is the self-pride property

• obviously Principle B effects need accounting for, etc.

D. <u>Functional NPs in General</u>

• once we have above mechanism for functional NPs, would expect to find them in other pla ces
• and, we do:

Unexpected Inference cases (discussed in a variety of literature):

(98) a. John (always) buys whatever/the thing that Bill buys.
b. $Bill_i$ buys his_i favorite car.
c. Therefore, $John_j$ buys his_j favorite car.

for details see Jacobson, L&P paper

basic idea:

John always z-buys the function f such that Bill z-buys f
Bill z-buys the function f: λx[x's favorite car]
hence:
John z-buys the function λx[x's favorite car]

• nothing extra is needed for these; it all comes out from what we have so far

is the same true in the standard theory, modulo the already-noted problem about the com ples translation of traces in functional questions and functional relatives ?
that is, are we just recycling arguments here:

no - there is still one additional piece needed in the standard theory

• Let *the thing that Bill buys* have a functional reading, as in von Stechow (1991) analysis

<u>But</u>- this is not good enough
<u>buys</u> does not want a functional argument, but an ordinary individual argument

(in variable-free - it type shifts by **z** so it's perfectly happy to directly get a functional argum ent here)

Hence: in addition - we need to supply a hidden variable as argument to the function *the thi ng that Bill buys* - in order to bind the argument of that function:

(99)

John(λx[x buys the function f which is the intersection of thing' and the s et of functions such that Bill is a y who buys f(y)) (x)]

a problem for either version of the story:

• functional readings on NPs are not possible anywhere

(100) a. John believes that the woman who every man loves just walked into the room.
b. The woman who every man loves is his mother.
--/--> c. $John_i$ believes that his_i mother just walked into the room.

the problem: • let *the woman who every man loves* be functional, and hence of category NP^{NP}
• it can combine with the VP just like a pronoun could (by **g**)
• *believes* can undergo **z** to have the subject position binding

<u>A note on the interaction of all of these cases with Weak Crossover</u>:

Interaction of functional questions with Weak Crossover (Engdahl, 1988; Chierchia, 1991 (based on observations of May):

(100) *Who loves every Englishman? His mother. (no functional reading)

Chierchia's story: let WCO crucially be a constraint on two actual things in a syntactic repr esentation
(a variable and binder, or an index on a trace and binder, or whatever)
then functional gap crucially needs to be represented as:
• $t_{f(x)}$ or f(x) etc. at relevant level
• in order for WCO to successfully block co-indexing (binding) between that and *e very Englishman* (or, the trace left by QR'ing *every Englishman)*

but given WCO story above, no need for this:

follows by having WCO a constraint on the combinatorics: this would require **s** on loves i n order to have the object slot "bind" into a functional argument in subject position
(see L&P paper and NALS paper for details)

s(loves') = λx[λf[loves'(x)(f(x))]]

• however we do wide scope quantification for objects in general would allow *every Englishman* to bind into a functional subject position
• but since **s** is disallowed, this can't happen

open question: May (85): this also has no pair-list reading. Does that follow from the Weak Crossover story? c.f., Chierchia, 1993

E. Other Functional NPs:

(i) Mitchell/Partee expressions

(101) a. Every man_i visited the $local_i$ bar
b. Every man_i visited the $local_j$ bar.

hence: "local" shows normal binding; can be bound or remain free

let *local* be of category $N^{NP}/_{R}N$ and of type <<e,t>,<e,<e,t>>
thus it maps a set of place-individuals (e.g., the *bar* set) into a function f from individuals to a set of place-individuals

NOTE: this means the superscript feature is not always a record of a pronoun inside an expression; it's really just a record of the semantic type

the local bar: **g** on *the* - maps an <e,<e,t>> into an <e,e>

"binding" of the argument position of the <e,e> function: done by **z** on visit'
every-man'(**z**-visit')(the-local-bar')

standard account: supply a hidden variable, in order to get the effect of binding
but note, no overt pronoun possible here:

(102) *Every man_i visited the local to him_i bar

Partee (1991): these show WCO effects:

(103) a. Every man_i was served at the $local_i$ bar.
b. *The $local_i$ bar served every man_i.

this follows without supplying a hidden variable; binding of the argument position of *the local bar* function would require **s** on serve

(ii) "Better Homes and Gardens" NPs:

• a variety of NPs (especially having to do with home and garden) easily shift into funcitonal readings:

(104) Every 19th century $landowner_i$ buried his grandmother in the $garden_i$.
(105) Everyone in $Berkeley_i$ puts eucalyptus on the $mantel_i$.

• same point again, no need to posit a variable as argument of the function

F. Across-the-Board Binding
(this has been noticed from time to time, cf., Dahl, 1981, Jacobson, 1984, Hohle, 1991, von Stechow, 1991, etc.)

(106) a. Every man_i loves and no man_j wants to marry $his_{i/j}$ mother.
b. Every man_i loves and no man_j marries $his_{i/j}$ mother.

Problems for direct compositionality given standard story:

- RNR constituent not in the right position to be bound
  - not c-commanded by the binder
- even if tried to scope out the binders, won't give the right reading:

(107) Every man loves or every man hates his mother (I can't remember which)
or has wide scope here

- most serious problem: one bindee - two binders, this makes no sense under standard vie w
  - hence, need some way to make extra copies of the bindee
  - two possibilities:
    - (a) reconstruction
    - (b) functional gap (von Stechow)

  will show that both have problems

<u>story under variable-free</u>: nothing new needed: follows immediately from Dowty/Steedman RNR plus va riable-free semantics:

(108) every man; $S/_R(S/_LNP)$; every-man'
loves; $(S/_LNP)/NP$; loves' $\text{--->}_{\mathbf{z}}$
loves; $(S/_LNP)/NP^{NP}$: $\lambda f[\lambda x[\text{loves}'(f(x))(x)]$
every man o loves
every man loves; $S/_RNP^{NP}$; every-man' o $\lambda f[\lambda x[\text{loves}'(f(x))(x)]$ =
$\lambda f[\text{every-man}'(\lambda x[\text{loves}'(f(x))(x)])$
(informally: the set of functions that every-man z-loves)
no man hates; similarly:
no man hates; $S/_RNP^{NP}$: $\lambda g[\text{no-man}'(\lambda y[\text{hates}'(g(y))(y)])$
(informally: the set of functions that no man z-hates)

every man loves and no man hates; $S/_RNP^{NP}$;
$\lambda f[\text{every-man}'(\lambda x[\text{loves}'(f(x))(x)]) \sqcap \lambda g[\text{no-man}'(\lambda y[\text{hates}'(g(y))(y)])$
$= \lambda f[\text{every-man}'(\lambda x[\text{loves}'(f(x))(x)]) \ \& \ \text{no-man}'(\lambda y[\text{hates}'(f(y))(y)])]$
(informally: the intersection of the set of functions that every man z-loves an d no man z-hates)

his mother; NP^{NP}; the-mother-of'

every man loves and no man hates his mother; S; the-mother-of function is in the ab ove intersection

<u>Two ways to do this with standard view of variables</u>:

a. reconstruction
b. functional gaps (von Stechow's proposal)

a potential problem for both:

• No binding out of just one conjunct unless there is binding out of both (cf., Hohle, 91; C hierchia, 87)

(109) *Every man_i loves but no man_j marries $his_{j/k}$ mother.

von Stechow: free pronouns different from bound pronouns - BUT:

(110) *Every man_k thinks that every man_i loves and (that) no man_j marries $his_{j/k}$ mother.

Note:
• Is this a more general problem? i.e., can we get the relevant reading in the full case?
• Does this have to do with stress?
• Does this have to do with pragmatics?
• Is this solved by looking at focus value of each conjunct?

(111) Every man_k thinks that every man_i loves his_k mother and that no man_j marr ies his_j mother.

• it may be more natural to stress the second occurrence of *his* which would mean that the t wo conjuncts aren't really "parallel", but it doesn't seem absolutely necessary
clearer case:

(112) Every man_i thinks that the bursar still has his_i paycheck but that every other man_j already deposited his_j paycheck. (stress on second *his* is not required)

(113) *Every man_i thinks that the bursar still has but that every other man_j already deposited $his_{j/i}$ paycheck. (M. Bittner)

Reconstruction:

(a) reconstruct, then index:
(114) a. Every man thinks that every man loves but (that) no man marries his mother ===> (reconstruct)
b. Every man thinks that every man loves his mother but (that) no man marries his mother ===> (index)
c. Every man_k thinks that every man_i loves his_k mother but (that) no man_j marries his_j mother.

(b) index, then reconstruct:
(115) a. Every man thinks that every man loves but (that) no man marries his mother ===> (index)
b. Every man_i thinks that every man_j loves but (that) no man_i marries his_i mother. ===> (reconstruct)
c. Every man_i thinks that every man_j loves his_i mother but (that) no man_i marries his_i mother.

attempt to block this: rule out LF in (c) by virtue of the fact that *every* man_i c-commands *no* man_i

Problem: This won't do any good for the case where the first occurrence of *his* remains free

Functional gaps: same problem!

(116) Every man λx[x thinks that

λf[every man (λy[y loves f(x)]) and (that) no man (λx[x loves f(x)])]

(λz[the-mother-of(z)])]

Generalization: No binding out of one conjunct unless there is binding out of both

Follows in variable-free:

(117) *... every man_i loves but no man_j marries $his_{k/j}$ mother

- *no man marries* - is looking for a functional argument - where the subject position of *marries* will bind the argument position of that function

  i.e., *marries* has undergone **z**

  no man marries; $S/_R NP^{NP}$; <<e,e>,t>

- *every man loves* is looking for a functional argument, but where it "passes the binding job" up

  i.e., *loves* has undergone **g**

  everyman loves; $S^{NP}/_R NP^{NP}$; <<e,e>,<e,t>>

- hence the categories (and the types) are not conjoinable

a striking prediction:

- recall that nested vs. crossed binding is just different order of applications of **g** and **z**
- **g**(**z**(think')) same semantic type as (**z**(**g**(think'))
  - any two expressions built with these two (in a parallel fashion) will also be of the same semantic type, though not the same meaning

    (since one wants to bind nestedly, and the other wants to bind crossedly)

hence, these should be able to conjoin (though this seems like a bizarre prediction)
and - they can!

(118)

Every man_i thinks that every boy_j should know but no boy_k wanted to hear any man_l say that $he_{i/l}$ would withhold $his_{j/k}$ allowance

(119)

Every man_i told his son_j but no boy_k wanted to hear his $father_l$ say that $he_{i/l}$ would withold $his_{j/k}$ allowance.

Some attempted solutions in the standard story:

(a) some kind of "parallelism" constraint

(b) focus condition (cf., Rooth, 1994 re VP Ellipsis)
- each conjunct must be in the focus value of the other

not clear how either of these could handle (118) and (119)
(though perhaps one could get that to work)

<u>Apparent (but not actual) problem</u>:

Conjuntion of pronoun-containing constituent and non-pronoun containing-constituent:

(120) Every man_i likes his_i dog and Mary's dog.

and; $(\alpha/\alpha)/\alpha$
so it can undergo **g** on an argument position (just like any other function)

z on *and:*

(121) Every boy_i and his_i mother came to the meeting.

G. <u>Paycheck readings of pronouns come for free</u>

Paycheck pronouns (= pronouns which show sloppy identity)

(121) The woman_i who put her_i paycheck in the bank was wiser than the one_j who put $\text{it}_{f(j)}$ in the Brown Employees' Credit Union.

(122) Every 3d grade boy_i loves his_i mother. Every 4th grade boy_j hates her_j.

<u>Two traditions using standard semantics with variables</u>:

(a) "Pronouns of Laziness" (pronoun has a full NP representation at LF) (Geach, 1962; Karttunen, 1969; Partee, 1974; Jacobson, 1977; Heim, 1991; ...)

cf: "e-type pronouns" though Evans (1977) actually meant something slightly different, and explicitly exempted paycheck pronouns from his use of this term

(123) every 3d grade boy (λx[x loves x's mother])
every 4th grade boy (λx[x loves <u>x's mother</u>])

her

(or, use different variables and have identity condition formulated in terms of alphabetic variance)

so: <u>x</u> is part of the meaning/LF representation of the pronoun, and is bound in the way that variable binding normally takes place
(either Binders Out, or Derived VP Rule)

(b) Free function variable approach (Cooper, 1979; Engdahl, 1986):

Let a pronoun (like a trace) correspond to a variable over functions of type <e,e> a
pplied to a variable over individuals:

(124) every 4th grade boy (λx[x loves the-mother-of'(f(x))])
f remains free, and picks up a contextually salient function (here, *the-mother*
-of function)

again: <u>x</u> is part of the meaning of the pronoun, and is bound in the way that variable
binding normally takes place, while <u>f</u> remains free

Engdahl: notes connection between this and functional questions

Solution here: will be a translation of Cooper/Engdahl approach into variable-free, and will show th
at once this translation is made, some problems with their particular implementation are solved

<u>Potential argument for the (b)-type approach</u> ("free function variable" approach)

should be a case of "deep anaphora" (where value of <u>f</u> can be contextually rather than lingui
stically supplied):

(125) new professor, standing in the mailroom waving her first paycheck:
What am I supposed to do with this?
Answer: Well, most of us usually put it in the bank.

note: occasionally it is claimed that these can't access functions from context:

(126) ?*Every married man thinks that she should cook dinner for him each night.

- but in fact, this is not impossible, and the availability of the relevant reading seems a bit "squishy"
- similar facts in the donkey literature:

(127) ?*Every donkey owner beats it.

<u>but</u>:
(128) Every Siberian husky owner needs to give it lots of exercize.

Q: why is it hard to get these with the function contextually supplied?
Tentative hypothesis: functions of type <e,e> are "fragile" objects - so they like to be made context
ually salient by being the meaning of some overt expression

Problems/questions/observations:

(a) <u>why</u> should pronouns have these extra meanings/LFs? Are they ambiguous in the lexicon?
- free function variable approach: a pronoun can either be <u>x</u> or <u>f(x)</u> - just accidental homop
hony
- pronouns of laziness approach - also accidental homophony

accidental homophony looks very suspicious, in view of the fact that the full set of pronouns has both ordinary and paycheck (sloppy) readings

note: there have been occasional attempts to answer this and to keep pronouns from just be
ing accidentally homophonous

but, as in the case of functional traces, these generally involve "generalizing to the worst case " - i.e., making the ordinary reading a special case of the paycheck reading

e.g., Engdahl: as with functional gaps - let "ordinary" meaning of a pronoun be a "0-place function"

-- another possibility would be to let it be a constant function\

– hence: take an ordinary pronoun which, in standard theory, is translated as x

let it instead be a paycheck pronoun f(y) but where f is a function which maps each individual into x (i.e., into the individual assigned to x on the relevant assignment function)

but all of these strategies involved generalizing to the worst case - and so predict that paycheck readings should be perfectly run-of-the-mill, which is not the case

(b) Engdahl: infinite number of meanings, how get these? (note: this is not a problem for the Pronouns of Laziness approach)

note: this exactly parallel to the functional question case

(129)

The woman$_i$ who told Sears$_j$ that the money she$_i$ owed them$_j$ was in the mail was wiser than the woman$_k$ who told Filene's$_l$ that it$_{W(k)(l)}$ hadn't been mailed yet

• so again: -- either pronoun is polymorphic (in the way a trace is) or there's a type-shift rule

<u>Claims</u>:
- In variable-free, the existence of paycheck readings for pronouns is automatic; it comes for free from the **g** rule (no lexical ambiguity)
- The infinite number of paycheck readings is also automatic (no extra type-shift rule needed - we already have it)
- The gender of the paycheck pronoun is immediately accounted for

to show this: translate free function variable approach into variable-free

will do this in two steps: step (1) eliminates the individual variable which is the argument of the function; step (2) eliminates the function variable

<u>Step 1</u>: • Take the paycheck pronoun to just be a simple free function variable
• Note: no need to supply an individual variable as argument of the function - because this argument "slot" will be bound by **z** rule
(same basic logic as in the case of functional questions)

(130) a. Every third grade boy loves his mother =
every-3d-grade-boy'(**z**(loves')(the-mother-of')
b. every fourth grade boy hates her =
every-4th-grade-boy'(**z**(hates')(f))

<u>problems with this</u>: (a) still requires lexical ambiguity -
meaning of ordinary pronoun is identity function over individuals; meaning of paycheck pronoun is free function variable
` (b) what about the additional paycheck meanings?
(c) new problem which other accounts didn't have: makes illegitimate use of a free variable (hence crucial use of variables)!!!

<u>step 2: get rid of crucial use of the function variable</u> (in the obvious way):

note: ordinary case - revise individual variable to be identity function over individuals
so obvious move here is to revise function variable to be identity function over functions (of type <e,e>)

thus: let pronouns also have the meaning λf[f] (identity function on functions of type <e,e>)

let this be of category $(NP^{NP})^{(NP^{NP})}$

(131) Every 4th grader hates her. informally:
hates first undergoes **z** - to be of type <<e,e>,<e,t>>
z(hates) wants an argument of type <e,e>, and then an individual argument
z(hates) then undergoes **g**. Both its argument slot <e,e> and its result category <e,t> are now changed so that each of them wants an argument of type <e,e>.
Hence: **g**(**z**(hate)) is of type < <<e,e>,<e,e>>, <<e,e>,<e,t>> >

The pronoun *her* is of type <<e,e>,<e,e>> (being the identity function on functions of type <e,e>; so can fill this slot.
hates her is thus of type: <<e,e>,<e,t>>

every 4th grader is of type <<e,t>,t>
It can undergoe **g** to be of type: < <<e,e>,<e,t>>, <<e,e>,t> >
and take *hates her* as argument

(this would be equivalent to function composing *every 4th grader* with *hates her*)

Every 4th grader hates her thus of type <<e,e>,t>. In order to compute extract propositional information, it is applied to some contextually salient functionof type <e,e>

In detail:

(132) hates; $(S/_LNP)/_RNP$; hates' $\text{--->}_{\mathbf{z}}$ hates; $(S/_LNP)/_RNP^{NP}$; $\lambda f[\lambda x[\text{hates'}(f(x))(x)]]$

$\text{--->}_{g_{NP}NP}$ hates; $(S/_LNP)^{NP^{NP}}/_R(NP^{NP})^{(NP^{NP})}$;

(for D a variable of type <<e,e>,<e,e>>)

$= \lambda D[\lambda g[\lambda f[\lambda x[\text{hates'}(f(x))(x)]](D(g))]] = \lambda D[\lambda g[\lambda x[\text{hates'}(D(g)(x))(x)]]]$

her; $(NP^{NP})^{(NP^{NP})}$; $\lambda f[f]$

hates her; $(S/_LNP)^{NP^{NP}}$; $\lambda g[\lambda x[\text{hates'}(g(x))(x)]]$

for simplicity, switch *every 4th grader* to *Bill:*

Bill; $S/_R(S/_LNP)$; $\lambda P[P(b)]$ $\text{--->}_{g_{NP}NP}$ Bill; $S^{(NP^{NP})}/_R(S/_LNP)^{NP^{NP}}$;

$\lambda V[\lambda f[\lambda P[P(b)](V(f))]]$ (for V a variable of type <<e,e>,<e,t>>) $=$ $\lambda V[\lambda f[V(f)(b)]]$

Bill hates her; $S^{(NP^{NP})}$; $\lambda f[\lambda g[\lambda x[\text{hates'}(g(x))(x)]](f)(b)] = \lambda f[\text{hates'}(f(b))(b)]$

• this solves problem (c) above - no crucial use of a variable
• But: once this done, problem (a) disappears!

Hepple's observation: The paycheck meaning $\underline{\lambda f[f]}$ is just **g** applied to the ordinary meaning.

(143) $\mathbf{g}(\text{her'}) = \mathbf{g}(\lambda x[x]) = \lambda f[\lambda y[\lambda x[x]](f(y)) = \lambda f[\lambda y[f(y)]] = \lambda f[f]$

Need to generalize earlier syntax: **g** rule above shifts A/B into A^C/B^C

Let it also shift A^B into $A^{C)^{(B^C)}}$

(possibly all rules should be generalized in the analogous way)

• Question in (b) also disappears: additional paycheck meanings: all just further applications of **g**

(144)

The woman_i who told Sears_j that the bill she owed them was in the mail was wiser than the woman_k who told Filene's_l that $\text{it}_{r(k)(l)}$ hadn't been mailed yet

it: identity function on functions of type <e,<e,e>>

it' (in lexicon): $\lambda x[x]$ (type <e,e>)

$\text{-->}_{\mathbf{g}}$ $\lambda f[f]$ (as given above) (type <<e,e>,<e,e>>)

$\text{-->}_{\mathbf{g}}$ (type < <e,<e,e>>, <e,<e,e>> > $\lambda V[\lambda x[\lambda f[f](V(x))]]$ (for V of type <e,<e,e>>)

$= \lambda V[\lambda x[V(x)]] = \lambda V[V]$

Hence:

- No accidental homophony. It follows immediately that pronouns should have paycheck meanings. This follows directly from the existence of the **g** rule.
  - Prediction: One would not expect to find a language with binding as in English, but with no paycheck readings for pronouns
- All additional paycheck meanings also a direct consequence of the binding system

Prediction: one could not find a language whose binding worked essentially the way that binding does in English but which does not allow paycheck (sloppy) readings for pronouns

The paycheck gender:

(145) John loves his mother. Bill hates her.

• why is the paycheck pronoun *her?*

Take free function variable approach + standard semantics of variables: (NOTE: there actually is n o problem under the Pronouns of Laziness approach)

(a) Assume gender in English is purely semantic:
(i) it is semantically predictable
(ii) it plays no role in agreement (only in pronominal system)

• then: meaning of *her* = f(x)

question: why should it be the range of the function which specifies the gender of the pro noun, rather than the argument of the function? (or, the domain of the function?)

her: meaning is a variable x over female individuals or
a complex variable f(x) for f a variable over functions whose range is the set of female indivi duals

(b) Suppose gender in English is also syntactic

(i) Give paycheck pronouns a simple representation in the syntax
[N (or, NP) [+Fem] her]

But why does this have the semantics f(x) where the +Fem feature determines the ra nge of x?

(ii) Give paycheck pronouns a complex representation in the syntax, whereby *he r* is the head

NP[+Fem]

N[+Fem] PP

her pro

(pick your favorite structure: the idea is to have a structure analogous to *the mother of Bill)*

her' : f pro: x (the two combine by functional application)

Problem: the "head" can be a variable over n-arguments and there can be any number of ar guments of this head
• Therefore, this is not a normal kind of head/complement structure

The gender in variable-free

A purely semantic account:
Notation: use w as a variable over female individuals, m as a variable over males, n as a variable over neuter individuals, x as a variable over the entire domain of individuals
and f also to represent the type of females; etc. - e to represent the type of individuals

Let *her* be the identity function on female individuals; hence λw[w] (and hence of type <w,w>
When it undergoes **g**, it will be of type <<e,w>,<e,w>>
i.e., the identity function on functions from individuals to females

• It thus follows immediately that the gender of the paycheck pronoun is a restriction on the range o f the function - this is because the range is what is inherited from the original meaning

If gender is syntactic: no problem either

her; $NP[F]^{NP[F]}$; λw[w] --> $_{\mathbf{g}}$ her; $(NP[F]^{NP})^{(NP[F]^{NP})}$; λF[λx[λw[w](F(x))]] (for F a v ariable over functions of type <e,w>) = λF[λx[F(x)]] = λF[F] i.e., identity function over functions of type <e,w>

H. i-within-i effects

(a) The complement of a relational noun cannot contain a pronoun "bound" by the head:

(35) a. *The/Every $wife_i$ of her_i childhood sweetheart came to the party.
b. *The/Every $wife_i$ of her_i childhood sweetheart's cousin came to the party.

(b) A genitive of either a relational or a regular noun cannot contain a pronoun "bound" b y the head.

the mystery: this doesn't seem to follow from anything deeply semantic (i.e., the attempted meaning is not incoherent)

(146) a. The/Every $woman_i$ who is married to her_i childhood sweetheart came to the party.
b. The/Every woman-
$_i$ who is married to her_i childhood sweetheart's cousin came to the party.

Other constituents which escape the violation:

(147) a. The/Every $woman_i$ marrying her_i childhood sweetheart next week came to the party.
b. The/Every $woman_i$ angry at her_i childhood sweetheart came to the party.
c. The/Every $woman_i$ killed by her_i childhood sweetheart became the subject of a made-for-tv movie.

• why can't *wife of her childhood sweetheart* have the meaning:
λx[wife'(x's childhood sweetheart)(x)]

in framework here: assume relational nouns <e,<e,t>> --->$_{\mathbf{z}}$ gives i-within-i violating meani ng:

(148) wife-of' --->**z** λf[λx[wife-of'(f(x))(x)]]
her-childhood-sweetheart' = the-childhood-sweetheart-of'
wife-of-her-childhood-sweetheart' = λx[wife-of'(the-childhood-sweethear t-of'(x))(x)]

Explanation: Nouns do not have syntactic subjects slots (atomic category N)
Nouns do not take subjects, even in Small Clause environments (unlike parti ciples, APs, etc.):
Hence: *table* of category N; *wife* of category $N/_{R}PP[OF]$

(149) a. With Sue marrying Bill next week, her parents' worst nightmare will become a reality.
b. With Sue angry at Bill, the party will be spoiled.
c. With Sue killed by Bill, the story will be made into a movie.

(150) a. *With that piece of wood table, we'll have plenty of room for eating.
b. *With Sue wife of Bill, he stands to inherit a lot of money.

(151) a. I would prefer Sue marrying Bill.
b. I would prefer Sue angry at Bill.
c. I would prefer Sue killed by Bill.

(152) a. *I would prefer that piece of wood table.
b. *I would prefer Sue wife of Bill.

Recall syntactic side of **z**:

- (B/NP)/A ---> $(B/NP)/A^{NP}$ (with possibility that NP should be replaced by a var iable over categories)
- this motivated by whole CG program (where syntax and semantics "mesh", and w here syntactic combinatorics regulate semantic combinatorics)

HENCE: relational nouns not of right syntactic category to undergo **z**

--> hence; wife; N/PP[OF]; λx[λy[wife-of'(x)(y)]]
semantics is right to undergo **z,** but syntax isn't

Compare to:

(153) The $woman_i$ who married her_i childhood sweetheart came to the party.

Here, *married* is of category (S/NP)/NP and undergoes **z**

(154) a. The $woman_i$ marrying her_i childhood next week sweetheart came to the party.
b. The $woman_i$ angry at her_i childhood sweetheart came to the party.
c. The $woman_i$ killed by her_i childhood sweetheart was mourned by all.

participles, APs, and passive VPs - all have syntactic subject slots: the above are thus syntac tically 2-place, and *married, angry (at), killed* can all undergo **z**

An Apparent Exceptions to i-within-i:

Karttunen/Nunberg generalization: the effect is lessened/goes away with transparent agentiv e nominals:

(155) a. *The/every $author_i$ of her_i mother's biography came to the party
b. ?The/every $writer_i$ of her_i mother's biography came to the party.

c. ?*Every author$_i$ of a best-seller about her$_i$ mother came to the party.
d. ?Every writer$_i$ of a best-seller about her$_i$ mother came to the party.

(156) ?The/every builder$_i$ of his$_i$ mother's house came to the party.

(157) a. ?Every lover$_i$ of his$_i$ mother's art collection will get to inherit it.
b. *Every lover$_i$ of his$_i$ mother's hairdresser will get to inherit many wigs.

• this follows: these are 2-place in the lexicon - can undergo **z** - then what's nominalized is **z**(write') etc.
• speaker variation/unrobustness of judgments: some speakers allow **z**'ed things to nominalize; others don't

Interactions of Paycheck pronouns, WCO, and i-within-i effects (Jacobson, 1977)

Jacobson (1977):

(a) the first pronoun in a Bach-Peters sentence is a paycheck pronoun; the second is an ordinary bound pronoun
(b) paycheck pronouns must have complex representations at LF containing an occurrence of the individual variable: because they show WCO effects and i-within-i effects as if they had complex representations
(Jacobson, 77 used Pros of Laziness approach; but Cooper/Engdahl does just as well as it contains the individual variable)

(158) The man_i who loves her_j kissed the $woman_j$ who wrote to him_i.

(159) a. the x: man'(x) and x loves f(x) [x kissed the y: woman'(y) and y loves x] (using Cooper/Engdahl)
b. the x: man'(x) and x loves the z : woman'(z) and z loves x [x kissed the y: woman'(y) and y loves x]

underlined portions in both correspond to the first pronoun

the claim that the first pronoun is a paycheck pronoun and the second a bound pronoun:
(i) (158) automatically has one such analysis, because paycheck pronouns can precede their antecedents

(159)
The woman who put it in the bank was wiser than the woman who deposited her paycheck in the Credit Union.

(ii) can't have the first be a bound pronoun and the second the paycheck pronoun, because that will violate WCO (object NP will have to bind the pronoun within the subject)

(iii) can't have both be bound pronouns without creating some kind of "Double Binding" representation (Keenan, 1971), or Absorption (Higginbotham and May)

(iv) can't have both be paycheck pronouns, as this recreates the original infinite regress problem

First pronoun shows WCO effects; second doesn't:

(160) a. The man_i who loves her_j kissed the $woman_j$ who wrote to him_i.
b. *The man_i who she_j loves kissed the $woman_j$ who wrote to him_i.

c. The man_i who loves her_j kissed the $woman_j$ who he_j wrote to.

First pronoun shows i-within-i effects; second doesn't:

(161) a. *His_i $wife_j$ kissed the man_i who loves her_j.

b. The man_i who loves her_j kissed his_i $wife_j$.

Jacobson (1977): explanation for all of these is that the first pronoun has a complex repres entation

Using Engdahl/Cooper:

(162) = representation (roughly) for (160b):
the x: man'(x) and f(x) loves t_x [x kissed the y: woman'(y) and y wrote to x]
cf.: *the man_i who the woman who wrote to him_i loves ___

(163) = rough representation for (161a):
the x: x is wife of f(x) gives i-within-i violation

These all follow in variable-free from combinatory constraints:

ordinary case:

(164) combinatorics for (161b):

λf[the man who **z**-loves f **z**-kissed the-wife-of function]

combinatorics for (160a):

λf[the man who **z**-loves f **z**-kissed λy[the woman who wrote to y]]

(165) = (160b): requires use of **s**:

λf[the man who f **s**-loves **z**-kissed λy[the woman who wrote to y]]

(166) = (161a): requires **z** on wife:

λf[the (**z**(wife-of')(f))] **z**-kissed λy[the woman who wrote to y]] =

λf[the [λg[λx[wife-of'(g(x))(x)]](f)] **z**-kissed λy[the woman who wrote to y]]

= λf[the λx[wife-of'(f(x))(x)] **z**-kissed λy[the woman who wrote to y]]

i.e., the unique x such that x is the wife of f(x) **z**-kissed the function mapping each y into
e woman who wrote to y

I. Some apparent exceptions to WCO (Lanik and Stowell, 1991)

Apparent Exceptions to Weak Crossover: Lasnik and Stowell

Case 1: the *tough*- construction

• well-known that *tough* gap has by and large the same properties as *wh*- movement gap (Chomsky, 1978 and others)

(a) domain separating *tough* adjective and gap is unbounded:

20. a. John is hard (for me) to imagine Mary wanting to invite __.
 b. John is hard (for me) to imagine Mary trying to persuade Sue to invite __.
 etc.

(b) parasitic gaps possible: (Maling and Zaenen)

21. John is hard (for me) to imagine friends of __ wanting to invite __.

(c) except in cases where gap is just one VP down, *tough* construction is an island:

Note : gap just one VP down - no island effect (Chomsky, 1978 and many others):

22. Which violin$_i$ is that sonata$_j$ easy to play $\_\__j$ on $\_\__i$?

But: gap further down - gives robust island effects (Jacobson, 1992):

23, a. It's hard to imagine John wanting to play that sonata on that violin.
 b. Which violin$_i$ is it hard to imagine John wanting to play that sonata on $\_\__i$?

c. That sonata$_j$ is hard to imagine John wanting to play __$_j$ on that violin.
d. *Which violin$_i$ is that sonata$_j$ hard to imagine John wanting to play __$_j$ on __$_i$?

Footnote: are some well-known differences between *tough* gap and run-of-the-mill *wh*- movement gap:

-- *tough* gap less happy in subject position:

24. a. Who do you imagine __ will be chosen?
b. *John is hard to imagine __ will be chosen.
c. ?*John is hard to imagine Mary claiming __ will be chosen.

-- *tough* gap less happy in tensed S:

25. a. Who do you imagine (that) Bill invited __?
b. ?John is hard to imagine (that) Bill invited __.

• general account of this (within GB, G/HPSG, and Categorial Grammar literature):

complement of the *tough* adjective has exactly/very similar syntactic structure and exactly/very similar semantics as material in a *wh*- construction

26. a. What do you **imagine Bill reading __?**
b. That book is hard (for me) to **imagine Bill reading __?**

Chomsky (1978), Browning (1984), etc: *tough* adjective takes a complement which has *wh*- movement within it, where what moves is an empty or silent operator

27. John is easy for me [$_{CP}$ *wh* $_i$ [PRO to please __$_i$]]

NOTE: *wh* or null operator must somehow be co-indexed with and/or bound by the subject

GPSG, HPSG, Categorial Grammar, etc. literature: *tough* adjective subcategorizes for a complement with a gap, as does a *wh* word
(Gazdar, 1981; Fodor, 1983; Jacobson, 1984; Hukari and Levine, 1990; Jacobson, 1992; etc.)

within Categorial Grammar (see Jacobson, 1992):

easy: takes as complement a $VP/_R NP$ (i.e., a VP with an NP gap on the right edge); hence, in (9), it's of category $((A/_L NP)/_R(VP/_R NP))/_R PP$

28. John is easy for me to please.

NOTE: as in above, need some way to establish semantic "linkage" between subject and gap position.

Jacobson, 1992: This nothing more than control, where control itself is a fact about lexical entailments.

In other words, *easy* takes 3 arguments: the PP - which denotes an individual, the VP/NP, which denotes a 2-place relation between individuals, and the subject, which denotes an indiv idual
It has associated with it some entailment such that for all individuals x and y and all relation s R, if *easy'(x)(R)(y)* then something is entailed about *x* (the PP argument) standing in the R-relation to *y* (the subject argument)
Hence, in (9) *easy'(me')(please')(j)* and so something is entailed about *please'(j)(me')*

• But: Weak Crossover Violations:

29. a. No man$_i$ is easy for his$_i$ mother to like __$_i$.
b. Who$_i$ is easy for his$_i$ mother to like __$_i$? (Lasnik and Stowell, 1991)

30. *Who$_i$ does his$_i$ mother like __$_i$?

31. a. No man$_i$ is easy (for me) to imagine his$_i$ mother liking __$_i$
b. Who$_i$ is easy (for you) to imagine his$_i$ mother hating __$_i$?

32. *Who$_i$/Which man$_i$ do you imagine his$_i$ mother liking __$_i$?

• Note: Regardless of the details of the analysis of the *tough*- construction, the bold portio n in (33) and (34) would appear to have exactly/essentially the same syntactic representatio n and the same meaning:

33. *Which man$_i$ do you imagine **his$_i$ mother liking __$_i$**?

34. No man$_i$ is easy (for me) to imagine **his$_i$ mother liking __$_i$**.

• If Weak Crossover is a constraint which holds within that domain, then something blocks the boldface domain in (14) from translating as:

x's mother likes x

The same principle should keep the boldface domain in (15) from translating this way

<u>Case 2: Parasitic gaps</u>:

35. Who$_i$/Which man$_i$ did you fire __$_i$ before **his$_i$ mother had a chance to warn __$_i$**?
(L&S, 1991)

36. *Which man$_i$ did **his$_i$ mother have a chance to warn __$_i$** ?

• Again, if Weak Crossover is a constraint holding within the bold-face domain, then somet hing blocks the boldface domain in (33) from translating as:

x's mother had a chance to warn x

The same principle should keep the boldface domain in (36) from translating this way

• Lasnik and Stowell's solution: The relevant constraint looks not only to the boldface doma in in the above, but also to the binder. If the binder is a "true quantificational" binder the co nstraint holds; if not the constraint doesn't hold.

<u>preliminary note</u>:

• <u>Binding into adjuncts</u>: (Every man_i left before his_i mother got here.)

Note: ***z*** binds a pronoun within some argument of a function to a higher argument of that function.
How do adjuncts? assume type-lifting - such that that *left* above (and all other arguments o f adjuncts) can type-lift to take the adjunct as argument.

This not actually necessary; one can get the same effect by applying ***z*** to *before.*

37. No man_i is easy (for me) to imagine **$\textbf{his}_\textbf{i}$ mother liking $\_\__\textbf{i}$**

--/--> (standard theory) x's mother like x

--/--> (variable-free theory) λx[x's mother like x] - this would be possible, but would re quire the use of ***s***(likes') - as in (24) above

• <u>the basic intuition</u>:

• the two do not "correspond to the same variable"; they are merged only later in the semanti c composition
---> λx[λy[x's mother likes y]]
• since the pronoun is not "bound" to the object slot, there is no Weak Crossover violation
• the pronoun is "bound" by the subject position of *easy* - via an application of ***z*** on *eas y*
• the subject position of *easy* moreover "controls" the object gap position, via lexical entail ments

<u>Case II</u>:

38. Who_i did you fire $\_\__i$ before **$\textbf{his}_\textbf{i}$ mother had a chance to warn $\_\__\textbf{i}$**?

--/--> (standard theory) x's mother had a chance to warn x

--/--> (variable-free theory) λx[x's mother had a chance to warn x] - this would be possible, but would require use of ***s***(warn')

• <u>the basic intuition</u>:
• again, the two do not "correspond to the same variable":
---> λx[λy[x's mother have a chance to warn y]]
• hence no Weak Crossover violation
• each gap bound separately, by the object position of *fire*

• <u>the details</u>:

(a) How can the object position of *fire* bind into the adjunct? If the adjunct is a VP modif ier, it will be introduced only <u>after</u> the object is introduced; hence the object is lower and so this itself should introduce a WCO violation.

<u>Assumption</u>: The *before* - clause is actually introduced before the object. Assume that it c an be a transitive verb modifier as well as a VP modifier.

39. a. I fired each man_i before his_i mother had a chance to warn him_i.
 b. Paul Masson will sell no wine_i before its_i time.
 c. Paul Masson will sell no wine_i before it_i has had a chance to properly age.

<u>Furthermore</u>: In this case we will treat the *before* clause as actually an argument - via type-lifting of *fire* to take the modifier as argument. (This follows from the general assumption: type-lifting to allow modifiers to be argument.) Recall, this not necessary; we could alterna tively do ***z*** on *before.*

Hence: *fire* takes 3 arguments: -- first, the *before* clause, then the object, then the subject
By ***z*** it can bind a pronoun within the *before* clause to the object slot

<u>NOTE</u>: These assumptions are independent of the framework here - analogous assumption s have been made/argued for in a variety of frameworks - see, e.g., Pesetsky (1994) and H ukari and Levine (1995).

(b) How do parasitic gaps?

Variety of ways; for convenience, pick modification of Steedman (1989)

Syntax: Suppose have a constituent which wants an argument C and a higher NP argument to give an expression of category A. Then it can shift to take a C with an NP "gap" to give an A with a "gap".

e.g., *fire* wants an adjunct and an object NP to give a VP. It shifts into something wanting an adjunct with an NP gap to give a VP with an NP gap. Thus it can combine with

before Mary had a chance to warn __

to give

fire __ before Mary had a chance to warn __

which is a VP containing an NP gap.

Semantics: The ***z*** rule!

<u>NOTE</u>: Syntactically this is slightly different from the ***z*** rule given in (20).
That rule has the effect that the expression takes a pronoun-containing constituent and a higher NP argument.
This rule has the effect that the expression takes a gap-containing constituent, and also has a "gap" instead of an overt NP object argument.

But semantically the two are identical; the pronoun and/or gap in the lower constituent is "merged" to the higher NP argument position.

(40) Who did Mary fire $\_\__i$ before Bill could warn $\_\__i$?

fire : takes *before* clause and then object to give a VP. Shifted *fire* takes *before* clause with a gap and no object and gives a VP; it "merges" the object argument position with the g ap position:

fire-__-before-Bill-could-warn-__' = λx[fire'(before-Bill-could-warn'(x))(x)]

Mary-fire-__-before-Bill-could-warn__' = λx[fire'(before-Bill-could-warn'(x))(x)(m)]

this proerty then occurs as argument of the question pronoun *who'*

(c) A question: this challenges the common wisdom that a parasitic gap cannot be c-comm anded by the other gap

something has to give with respect to this assumption or with respect to Weak Cro ssover violations in <u>any</u> theory

Other cases of object gaps supporting other "lower" gaps:

(41) a. ?Which man_i did you persuade $\_\__i$ to ask Mary to invite $\_\__i$?
b. ?Which man_i did you tell $\_\__i$ that you would vote for $\_\__i$? (Engdahl, 1984)

So, the generalization would seem to be only that a gap in <u>subject</u> position can't suppor t
a lower parasitic gap

• Putting this together:

42. Who_i did you fire $\_\__i$ before his_i mother had a chance to warn $\_\__i$?

before-his-mother-had-a-chance-to-warn' =
λx[λy[before x's mother have a chance to warn y]]

fire' undergoes 2 applications of ***z*** .
One is the normal ***z*** rule, which allows it to take a *before* clause with a pronoun within it, a nd "binds" that pronoun to the object slot.
The other occurrence of ***z*** is induced by the "parasitic gaps" rule, which allows it to take a *before* clause with a gap (and then to be "missing" its object), and it "binds" that gap pos ition to the object slot.

The pronoun and the gap are thus ultimately "merged", even though they are not "merged" i n the meaning of *his mother had a chance to warn __*.