Direct Compositionality and Variable Free Semantics
ESSLLI 13 - Helsinki
August, 2001

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

Course Outline (Tentative):

Session 1: Background; Motivation for direct compositionality; initial motivation for vartiable-free semntics; mechanics of variable-free semantics and of direct compositonality

Sessions 2 - 4: Some initial payoffs: functional readings in a variety of constructionss; binding and coordination; paycheck pronouns; weak crossover (and i-within-i effects) and some apparent exceptions to weak crossover

Sessions 4 and 5: Further applications: Direct compositional analyses of ellipsis, Pied-Piping, and focus; the interactions of focus and paycheck pronouns; variable-free treatments of focus. Time permitting, will end with a discussion of open problems and challenges to the direct compositional approach.

NOTE: It is very unlikely that we will actually cover all of these topics. Most are covered in the accompanying handouts and in the readings; we will thus probably pick some subset of the above topics to go over in some depth.

Readings:

The following papers of mine deal with variable-free semantics, and will be available on the
      course web page:

"Binding Connectivity in Copuar Sentences",in *Proceedings of the 4th Conference on Semantics and Linguistic Theory,* 1994. Distributed by Cornell Working Papers in Linguistics.

The Locality of Interpretation: The Case of Binding and Coordination",   in *Proceedings of the 6th Conference on Semantics and Linguistic Theory,* 1996.   Distributed by Cornell University Working Papers in Linguistics.

"Towards a Variable-Free Semantics", *Linguistics and Philosophy* **22***,* 1999. pp. 117-184.

"Paycheck Pronouns, Bach-Peters Sentences, and Variable-Free Semantics", in *Natural Language Semanticss* **8.2***,*   2000. pp. 77-155.

"Antecedent Contained Deletion and Pied-Piping: Evidence for a Variable-Free Semantics", in *Proceedings of t he Eighth Conference on Semantics and Linguistic Theory,* 1998. Distributed by Cornell University Working Papers in Linguistics.

"Paychecks and Stress", in *Proceedings of the Tenth Conference on Semantics and Linguistic Theory,* 2000. Distributed by Cornell Working Papers in Linguistics, (18 pp.)

Someone wishing to read some of these can take one of three routes:

Variable Free Light: Variable-free light: this would involve reading mainly the SALT 6 paper which is a nice, easy introduction to the program, plus maybe SALT 4.

Variable-free non-light: either the *Linguistics & Philosophy* paper and/or the *Natural Language Semantics* paper. If someone were to read one of these, then the second would become much easier as there is considerable overlap. Either order is possible.

Variable-Free extensions: The SALT 8 and SALT 10 papers. SALT 8 is on Antecedent Contained Deletion, focus, and var-free; SALT 10 is on focussed pronouns and paycheck pronouns.

Some Additional Useful Readings:

Background:

Jacobson, P., "The Syntax/Semantics Interface in Categorial Grammar" in S. Lappin (ed.), Handbook of Contemporary Semantic Theory, Blackwell, 1996.

Dowty, D., "Type-Raising, Functional Composition, and Non-Constituent Conjunction", in R. Oehrle, E. Bach, and D. Wheeler (eds.), Categorial Grammars and Natural Language Structures, Reidel (Studies in Linguistics and Philosophy series), 1988.

Steedman, M. "Combinators and Grammars", in R. Oehrle, E. Bach, and D. Wheeler (eds.),Categorial Grammars and Natural Language Structures, Reidel (Studies in Linguistics and Philosophy series), 1988.

Jacobson, P., "Flexible Categorial Grammars: Questions and Prospects" in R. Levine (ed.), Formal Grammar: Theory and Implementation, Oxford University Press (Vancouver Studies in Cognitive Science), 1992.

Challenges:

Heim, I., "Predicates or Formulas? Evidence from Ellipsis", in A. Lawson and E. Cho (eds.), *Proceedings of the Seventh Conference on Semantics and Linguistic Theory* (Stanford),Cornell Working Papers in Linguistics, 1997.

Sauerland, U., *The Meaning of Chains,* Chapter 4. MIT Dissertation, 1998. Distributed by MIT Working Papers in Linguistics.

Dimitriadis, A. "Function Domins in a Variable Free Semantics", to apper in *Proceedings of SALT 11*.

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. Program of Direct Compositionality and a little bit of Categorial Grammar apparatus

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)
                  Additional structure necessary: 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 directional features (except in exceptional cases such as postpositions in English - e.g., *ago)* and other features (case selection features where case is not lexically specified), etc.

Syntax/semantics "interface":

- Semantic type predictable from syntactic category

     each syntactic category corresponds to some semantic type
     any expression of category A/B is of type <b,a>, for b the semantic type of
          expressions 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

Generalized binary rule schemata

(1)   Right Concat:
          Let   be an expression: <[ ]; A/$_R$B;   '> and   be an expression:
          <[ ]; B; '>. Then there is an expression : <[   ]; A;   '(')>

(2)   Left Concat:
          Let   be an expression: <[ ]; A/$_l$B;   '> and   be an expression:
          <[ ]; B; '>. Then there is an expression : <[   ]; A;   '(')>

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

(4)   Infixation:
          Let   be an expression: <[ ]; A/$_l$B;   '> and   be an expression:
          <[x | y]; B; '>. Then there is an expression : <[x [ ] y]; A;   '(') >

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

Adding in some unary rules
          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 type; 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 $Lift_b(x) = f[f(x)]$
hence $Lift_b(x)$ is a function $(a\text{-->}b)\text{-->}b$

Generalized Lift rule:

(5) $\underline{Lift_B}$
        1. Let   be an expression $<[\ \ ]; A;\ \ '>$. Then there is an expression   :

        $<[\ \ ]; B/_R(B/_LA);\ lift_b(\ ')>$

                Abbreviation: $\mathbf{l_b}(\ )$ will be used to abbreviate the full triple

        2. Let   be an expression $<[\ \ ]; A;\ \ '>$. Then there is an expression   :

        $<[\ \ ]; B/_L(B/_RA);\ lift_b(\ ')>$

        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; x[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   be an expression: $<[\ ];\ /\ _RB;\ '>$ and   be an expression:

        $<[\ ]; B/_RC;\ '>$. Then there is an expression   : $<[\ ];\ /\ _RC;\ '\ o\ '>$
        <u>Left composition:</u>

Let   be an expression: <[ ]; / $_L$B; '> and   be an expression:
<[ ]; B/$_L$C; '>. Then there is an expression : <[ ]; / $_L$C; ' o '>

Notes: (1) Steedman proposes two additional cases of "mixed composition" but
where these apply only with certain categories; not clear whether these are
needed (depends also on how work this in with Wrap operations)
Again: this can be generalized if categories are taken to be operations on
strings
(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;
known as the "Geach" rule, **g**, or "Division"
• thus **g** maps a function f into something which then wants to take as argument a
function h such that it returns the value one would have gotten by function
composing f with h
i.e., **g**(f) (h) = f o h

Recasting in this way:

(7)   Let   be an expression <[ ]; A/$_R$B; '>. Then there is an expression   of the
form: <[ ]; (A/$_R$C)/(B/$_R$C);   V[c['(V(c))]]>
note the semantics here could equivalently have been written:
V[' o V]
So - something which wants a B to its right maps into something which
wants an "incomplete B" - a B/$_R$C to its right - and it "inherits" the
information that a C is still missing (both in the syntax and in the semantics)

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

A quick tour of some previous applicatons

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]; (/$_L$)/$_R$;   > for   any category whose final result is S
where   = & for two objects of type t, and otherwise
=   f[g[x[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/$_L$S)/$_R$S ; &>
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
are instantiated by general principles of directionality in English:

X/X ---> X/$_L$X	(modifiers go on the left)
S/X ---> S/$_L$X	(subjects of S go on left)
A/B ---> A/$_R$B	(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'   --->$_{lVP}$ beans; VP/$_L$(VP/$_R$NP);   R[R(beans')]

("VP" = S/$_L$NP)

yesterday; VP/$_L$VP; yesterday'

combine these by left composition:

beans yesterday; VP/$_L$(VP/$_R$NP); yesterday' o   R[R(beans')]   =

R[yesterday'(R(beans'))]

similarly: peas today; VP/$_L$(VP/$_R$NP):   S[today'(S(peas'))]

beans yesterday and peas today; VP/$_L$(VP/$_R$NP);

R[yesterday'(R(beans'))]   S[today'(S(peas'))] =

T[R[yesterday'(R(beans'))](T)   S[today'(S(peas'))](T)] =

T[yesterday'(T(beans'))   today'(T(peas'))]

ate; VP/$_R$NP; ate'

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

Right Node Raising:

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

Mary; NP; m --->$_{lS}$ Mary; S/$_R$(S/$_L$NP):   P[P(m)]

loves; (S/$_L$NP)/$_R$NP; loves'

Mary loves; S/$_R$NP;   P[P(m)] o loves' =   x[P[P(m)](loves'(x))] =   x[loves'(x)(m)]

similarly;   John hates; S/$_R$NP;   y[hates'(y)(j)]

Mary loves and John hates; S/$_R$NP;   x[loves'(x)(m)]   y[hates'(y)(j)] =

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

a (probably oversimplified) account of relative clauses

- let a relative pronoun like *who(m)* be listed as:

$$<[who(m)]; (N/_LN)/_R(S/_RNP); \quad P[\ Q[\ x[P(x)\ \&\ Q(x)]]]$$
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
   later that this is not correct

(11) every woman who Bill invited

invite; $(S/_LNP)/_RNP$; invite'
Bill; NP; b --->$_I$ Bill; $S/_R(S/_LNP)$;   P[P(b)]

function compose these two:

Bill invited; $S/_RNP$; x[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; $N/_LN$;   Q[x[y[invite'(y)(b)](x) & Q(x)]] =
        Q[x[invite'(x)(b)] & Q(x)]]

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

every woman who Bill invited; $S/_L(S/_RNP)$;   P[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
  program because:
  - standard account = Binders Out approach (Montague 1973 - Quantifying-In;
  Lakoff, 1970, McCawlye, 1972 - Quantifier Lowering; May, 1976 - Quantifier
  Raising)
    - these all posit a level of representation at which an object quantificational
  NPs may be out of the sentence; where a variable (and/or indexed pronoun) is in its
  place, where this variable is -abstracted over, and where the whole thing is then
  applied 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 mechanisms 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 ambiguities

     <u>some in situe approaches</u>:
          (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 quite as much as does Hendriks, and gets scope ambiguities by a combination of function composition and argument position lifts
          (c) Steedman (2000)
          (d) Barker (2001) - uses "continuations" techniques (a kind of generalized lifting???)

- issues are complicated by the fact that the data is complex (see, e.g., Szabolcsi, etc. - on the 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 (and 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 composition to proceed

Note that: (1) follows from (ii) but not vice-versa; (i) is compatible with the idea that functions 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

<u>invite'</u> - 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*
       argument 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 levle
            (b) post 1975 - version: posit a surface trace in this position, which translates as a
            variable <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
            representation   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
                assignment 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</u>
                <u>linguistic expression</u>   has as its meaning a function from the set of
                assignment functions 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
        individuals)

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

            x[b invite x]

        • various possiblities here of how/where the   gets put into the tree: one slogan is
        "movement creates a   " - but one needs to spell out just what this means -

when/where in the derivation do we actually put in the λ?; another possibility is to take the *who* to map into _x 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:

　　　　λg[woman'(g) ∩ λx[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:
　　　　[[woman who b invited t]]^g = [[woman]]^g ∩ [[λx[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 abstract 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
        variable-free

the key: we don't stick a variable in object position to allow functional application to
        happen and then later   -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:

                x[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
        altogether

III. The variable-free program in a nutshell

A. FIRST: SPELLING OUT THE STANDARD THEORY

• as noted above:
        — model-theoretic apparatus includes an additional "layer" - the set of assignment
        functions G
        -- where each assignment function g in G is a function from each variable name to
        some 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: $[[walks]]^g$ $([[Bill]]^g)$

                or: g[walks(g)(Bill(g))]
        note: for convenience I am taking Bill *here to denote an individual rather than a
        generalized 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 x, and use y 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
        same
• 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 $\underline{x}$ 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'   --->   var[VP'(var)]   (for var a variable over variables)

        • may also be coupled with a syntactic bit; see below
        • this particular formulation allows for vacuous   -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 $\underline{x_{42}}$; 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   be a triple of the form: <[ ]; VP [IND: M];   ', for i a member of M>.

        Then there is an expression   of the form: <[ ]; VP [IND: M-i];   x$_i$['(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 $\underline{x}$
                            • 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)) --->   x[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 $(S/_LNP)/_WNP$
        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   -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

<u>C. A Brief discussion of each of these</u>

• apparent advantage of Binders Out: uses same mechanism for binding as can be used for
     quantifier scopes; the pulling out of the binder and   -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 <u>slot</u> as this is built into the rule
     • well-known problems: this inherits the problems of a c-command account,
     including *Every man's mother loves him* - see much literature on this including,
     most recently, 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
     two variables at LF with the same name
     • Strictly speaking, though, this not quite true of the derived (T)VP approach: this
     is a relationship between an argument <u>slot</u> and a variable/trace/pronoun in some
     syntactic 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. <u>Variable-Free Semantics</u>   (as done in my papers; other versions are possible and have
     been proposed; will return)

Basic ideas:
     • no variables in the semantics, no assignment functions (variables for notational
     convenience only)
     • all meanings are good, healthy model-theoretic objects
     • an expression C which contains a pronoun unbound within C denotes a function
     from 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
     whatever
     • pronoun itself then must denote a function from individuals to individuals - in
     particular 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: x[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]

standard story: lowest S is same semantic type in all three cases (a function from
   assignment functions to propositions)
variable-free: lowest S has different semantic type in all three cases:
    (a) = $\langle e,\langle e,t\rangle\rangle$
    (b) = $\langle e,t\rangle$
    (c) = t

apparent advantage of the standard story:
   • 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
   assignment functions to something - the semantic combinatorics is uniform
   throughout, and 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
   same distribution (modulo the distribution of resumptive pronouns).
      But, stay tuned......

the meaning of the pronoun:
   he' = identity function on individuals =   x[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 x[x] =   x[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 $\langle e,e\rangle$:
   x[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) =   λf[λx[h(f(x))(x)]]   (for f of type <e,a>)

Intuition:
        <u>love'</u> is an ordinary relation between two individuals (of type <e,<e,t>>)
        <u>*z*(love')</u> 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 <u>the-mother-of'</u>
        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') =   λx[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):

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

        he lost has the meaning <u>lost'</u>
        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') =   λx[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 pronouns are allowed in certain places where expressions without them are not sanctioned

(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   x[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 combinatorics
          • 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 turn will provide an automatic account of i-within-i effects

Hence: 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 expression be an expression of category A^{NP}

Pronoun: in lexicon: NP^{NP}
<[he]; NP^{NP};   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: $A/B \longrightarrow A^{NP}/B^{NP}$

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 $\boldsymbol{g}(f) = $ V[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 $\sim$
  We can map this into a VP operator *not*   by "Geaching" it:
    not' $= $ P[x[$\sim$P(x)]]

Another observation: "Geach" is a unary (Curry'ed) version of function composition
  Thus $[\boldsymbol{g}(f)](h) = $ f o h

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

(29)   the **g** rule:
  Let   be an expression: $<[\ \ ]; A/B;\ \ '>$. Then there is an expression   :
    $<[\ \ ];\ \ ^C/B^C;\ \boldsymbol{g}_c(\ ')>$

(30)   Example: composition of *he lost*

lost; $S/_LNP$; lost' $\text{---}>_{\boldsymbol{g}}$ lost; $S^{NP}/_LNP^{NP}$;   f[x[lost'(f(x))]]
he; NP^{NP};   y[y]
he lost; S^{NP};   f[x[lost'(f(x))]](y[y])   (NOTE: by definition of **g** this will give the
                              same result as if we'd function-composed
                              lost' with y[y])
      $= $ x[lost'(y[y](x))] $= $ 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' $\text{---}>_{\boldsymbol{g}}$ loves; $(S/_LNP)^{NP}/_RNP^{NP}$;   f[x[loves'(f(x))]]
him; NP^{NP};   y[y]
loves him; $(S/_LNP)^{NP}$;   f[x[loves'(f(x))]](y[y]) $= $ x[loves'(y[y](x))] $= $
        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</u>
      <u>its syntactic category</u>

$-->_l$
Mary; $S/_R(S/_LNP)$; $P[P(m)]$

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

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

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

Giving the syntax for the z rule:

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

Semantics is the same as above; the difference is that this effects syntactic category in the
        predictable 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) $z_B(((B/NP)/...A) = ((B/NP)...)/A^{NP}$
        Given a function f of type $<a,<d_1, ...,d_n<e,b>>>$, z_b (f) is a function of type
        $<<e,a>,<d_1, ..., d_n<e,b>>>$, where $z_b(f) =$ $G[$ $D_1, ...,D_n[$ $x[$ $f[f(G(x))(D_1), ..., (D_n)](x)]]]$
        (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
        general 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
        restriction. 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} and 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
        involve binding of a kind of VP anaphor (not all can, since it happens across
        sentences, 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 expression 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 expressions 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 environments; 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: listener computes a proposition (since this how information is conveyed) 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 functions 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 to 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, -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-passing 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, functional 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 conditions 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 names

- Some common objections:

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

Answers:

(i)   but there are lots of cases where there are binding effects without overt variables/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 readings)

(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
        transparent rendering of the surface syntax

IV.   *Preview of some of the advantages and also of how this connects up to direct
        compositionality*

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
        (roughly, binder must c-command bindee)
•   but on the surface there are lots of cases where this doesn't hold; where "small"
        expressions stand alone, yet show binding effects
• i.e., binders without bindees in their c-command domain
• bindees without c-commanding binders

hence: posit a level of representaiton at which things moved around and/or extra copies of
        things 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
        NP, and it affects the semantic type of every expression that it is in "all the way up"
        until 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
            undergo **z**
            z(loves) function composes with *no man*
            function composition all the way up, gives:

                        f[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'(x[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:
          $A/B \; ---> \; A^{NP}/B^{NP}$
   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 modifier - to take that as argument.

 Example, in a case not involving binding:

 (44)   John saw Bill yesterday.

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

(45)   see Bill; VP; see'(b) $--->_l$ see Bill; $VP/_R(VP/_L VP)$;   f[f(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 passing up pronoun features (along with the appropriate semantics). But as long as lift is independently motivated and hence needed anyway, there is no need to do anything new for these cases.

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
          sense in which it's correct to say that *he* binds *his* - binding is a relation between
          argument slots - so its more appropriate to say that *his* is bound by the subject slot
          of *feed.*

Case 2: This is the more interesting one. What about cases where one pronoun cannot
"bind" 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:
                      x[y[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'   --->$_z$ think; (S/NP)/S^{NP};   P[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

               -->$_z$ think; ((S/NP)/S^{NP})^{NP};   R[y[z(think')(R(y))(y)]] =
                                       R[y[P[x[think'(P(x))(x)]](R(y))(y)]] =
                                       R[y[x[think'(R(y)(x))(x)](y)]] =
                                       R[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 compositon: this will have a payoff in that it will give an
         automatic account of Lasnik 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 man$_i$ thinks that every boy$_j$ said that his$_j$ mother loves his$_i$ dog.
(55)   Every man$_i$ thinks that every boy$_j$ said that his$_i$ mother loves his$_j$ dog.

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

say; (S/$_L$NP)/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;
        the outermost argument position will be "passed up" for higher binding   gives (16)

say; (S/$_L$NP)/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
                        -->$_z$ (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;
        the 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')(z_t(g(said'))(his-mother-loves-his-dog'))))

Excercize: work through the full derivations. One is done in my *L&P* paper; work through the other 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: only 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; it 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 binding 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) =   x[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-command) rather than left-to-right order. Could be restated using left-to-right order (since 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-command", "f-command" - take your pick); if linear order turns out to be right it might be 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 <u>a constraint on the relationship between two actual things sitting in some level of syntactic representation</u> (a pronoun/trace/variable and a binder)

      (a) ultimately, aconstraint on rules (Postal, 1971; Jacobson, 1972, 1977; vs. (b) a constraint 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; Jacobson, 1972, 77; etc.) vs. (b) c-command (Reinhart, 1983 and most things since)

      here: <u>constraint on the combinatorics</u>: binding is only between a higher argument position 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 order to get binders and bindees to be in the right configuration to express certain WCO generalizations; here no need for such extra stuff since the constraint constrains the combinatorics of "merging" argument slots

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

V. <u>Some Initial Payoffs</u>

A. <u>Functional Questions</u>   (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 arguments

• 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
        for questions also give wrong reading using wide scope reading for *no Englishman*

<u>solution of Groenendijk and Stofkof (83) and Engdahl (86)</u>:   (basic semantics of this
        solution 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' (x[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
        exactly 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)
                                ---> -abstraction over y

                                                y[love'(x)(y)]
                                        this then is argument of every-man'
                                        which is equivalent to above
                                in non-Binders Out approach: directly take the
                        meaning of the VP as argument of the subject (cf.,
                        Montague, 73)

assume that x is then   -abstracted over:

  x[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
analysis of the meaning of *who*

Comment: • this much like the point made earlier about the standard way to do the
        semantics of relative clauses
        • first we posit a hidden variable in object position to get the semantics
        combinatorics to only involve functional application
        • then later we   -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
        question) 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*
        position and the extraction position at surface structure; G&S do things slightly
        differently; a third implementation given in Chierchia (1991) who posits complex
        indices on the traces in the surface syntax

(67) $t_{f(x)}$ ---> f(x)   (for notational convenience, I adopt something like complex
                                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   -abstract over x   --->   x[loves'(f(x))(x)]
                then take this as argument of every-man'
                                every-man'(x[loves'(f(x))(x)])

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

        • next step: recall that in ordinary questions, somewhere along the line the x
        variable which corresponds to the object trace gets   -abstracted over

        • so, in functional questions, the f variable will then get   -abstracted over by
        presumably the same process

        --->   f[every-man'(x[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) has to be polymorphic

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

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 $\langle e,e \rangle$
> • but, can have questions about n-place functions
> > (functions of type $\langle e,\langle e,e\rangle\rangle$ 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 $\langle e,\langle e,e\rangle\rangle$

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' o love' =
> > > x[every-Englishman'(love'(x))]

- recall: we don't need to posit a variable which later gets -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 x̲)
- for functional questions:
    - Why̲ should it also have this additional, complex meaning?

- NOTE̲: The main point: functional readings are unexpected; we need to posit something extra to get them (allow the trace, or whatever, to have the complex̲ meaning f̲(̲x̲)̲ as well as the simple meaning x̲)

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 f̲(̲x̲)̲ 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
        position 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
        unexpected
        crucially, this arises from the assumption that "binding" effect requires an actual
syntactic 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 functions 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 nothing
        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
        the same way that multiple pronouns are allowed in general)

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

love; $(S/_LNP)/_RNP$; loves' $-->_z$ love; $(S/_LNP)/_RNP^{NP}$;   f[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; $S/_R(S/_LNP)$; every-man'

every man loves; $S/_RNP^{NP}$; every man o **z**(love) =
every man love; $S/_RNP^{NP}$;   f[every-man'(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
        proceeds 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 $S/_RNP^{NP}$ as argument as well as an $S/_RNP$

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

                $<<e_{3(i)}, e_1>,<e_{4(i)},<e_2,t>>>$ $--->_z$
                $<<e_{5(i)}, <e_{3(i)}, e_1>>,<e_{4(i)},<e_2,t>>>$

Answer: (i) the syntax of the **z** rule will stop this: things are bound to argument slots which are actual syntactic argument slots of the verb; things are not bound to semantic argument slots which syntactically come out as superscripts

(ii) actually, I'm not even sure that any bad result would happen if one allowed 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 surprising would be if we didn't find these
- on the other hand, they require extra type-shifts, and so could lead to processing difficulties (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; $S/_RNP$;   x[invite'(x)(q)] --->$_g$

        the queen invite; $S^{NP}/_RNP^{NP}$;   f[y[x[invite'(x)(q)](f(y))]] =

                f[y[invite'(f(y))(q)]]

        z(think); $(S/_LNP)/S^{NP}$;   P[x[think'(P(x))(x)]]

        z(think) the queen invite; $(S/_LNP)/_RNP^{NP}$;

            P[x[think'(P(x))(x)]] o f[y[invite'(f(y))(q)]] =

            f[P[x[think'(P(x))(x)]](y[invite'(f(y))(q)])] =

            f[x[think'(y[invite'(f(y))(q)](x))(x)]] =

            f[x[think'(invite'(f(x))(q))(x)]]

          `   (informally: f[x[x thinks q will invite f(x)]]

        every man; $S/_R(S/_LNP)$; every-man'   this composes with above:

        every man thinks the queen invite; $S/_RNP^{NP}$;

            every-man' o   f[x[think'(invite'(f(x))(q))(x)]] =

            f[every-man'(x[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
    any 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
    account needs an extra bit:
        trace after *believe* can translate as a function of type <e,p> applied to an
    individual 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
    one'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 --->$_{z}$ loves; $(S/_{L}NP)/_{R}NP^{NP}$;   f[x[loves'(f(x))(x)]]
        every Englishman; $S/_{R}(S/_{L}NP)$; every-Englishman'
        every Englishman o loves; $S/_{R}NP^{NP}$;   f[every-Englishman'(x[x 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., = x[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   -abstract over the open variable in the meaning of *his mother:*

his mother; x's mother   --->   x[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

A note on connectivity effects in copular Ss in general:

- 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 connectivity 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
       on 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   x[proud-of'(x)(x)]]

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

• obviously Principle B effects need accounting for, etc.

D. Functional NPs in General

• once we have above mechanism for functional NPs, would expect to find them in other
       places
• 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: x[x's favorite car]
       hence:
       John z-buys the function   x[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
       comples 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

But- this is not good enough
buys 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
       argument here)

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

(99)   John(x[x buys   the function f which is the intersection of thing' and the set 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

A note on the interaction of all of these cases with Weak Crossover:

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 representation
                (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 *every 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 in 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') =   x[f[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. <u>Across-the-Board Binding</u>
        (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
          view
          • 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

story under variable-free: nothing new needed: follows immediately from Dowty/Steedman RNR plus
          variable-free semantics:

(108)   every man; $S/_R(S/_LNP)$; every-man'
          loves; $(S/_LNP)/NP$; loves'   --->$_z$
          loves; $(S/_LNP)/NP^{NP}$:   f[x[loves'(f(x))(x)]
          every man o loves

                  every man loves; $S/_RNP^{NP}$; every-man' o   f[x[loves'(f(x))(x)] =
                          f[every-man'(x[loves'(f(x))(x)])
                  (informally: the set of functions that every-man z-loves)
          no man hates; similarly:

                  no man hates; $S/_RNP^{NP}$:   g[no-man'(y[hates'(g(y))(y)])
                  (informally: the set of functions that no man z-hates)

          every man loves and no man hates; $S/_RNP^{NP}$;
                    f[every-man'(x[loves'(f(x))(x)])   g[no-man'(y[hates'(g(y))(y)])

                =   f[every-man'(x[loves'(f(x))(x)]) & no-man'(y[hates'(f(y))(y)])]
                    (informally: the intersection of the set of functions that every man z-loves
                          and 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
                    above intersection

Two ways to do this with standard view of variables:

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;
          Chierchia, 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$_j$ marries his$_j$ mother.

• it may be more natural to stress the second occurrence of *his* which would mean that the two 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.

         <u>attempt to block this</u>: rule out LF in (c) by virtue of the fact that *every man$_i$*   c-
             commands *no man$_i$*
         <u>Problem</u>: This won't do any good for the case where the first occurrence of *his*
             remains free

Functional gaps: same problem!
(116)   Every man   x[x thinks that

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

                     (z[the-mother-of(z)])]

<u>Generalization</u>: 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

<u>a striking prediction:</u>

• 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.

<u>Some attempted solutions in the standard story:</u>

(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; (/)/
> 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.   Paycheck readings of pronouns come for free

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 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$_j$.

Two traditions using standard semantics with variables:

(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 (x[x loves x's mother])

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

<div style="text-align:center">

her

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

</div>

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> applied to a variable over individuals:

(124)   every 4th grade boy (x[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 that once this translation is made, some problems with their particular implementation are solved

Potential argument for the (b)-type approach ("free function variable" approach)

should be a case of "deep anaphora" (where value of $\underline{f}$ can be contextually rather than linguistically 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.

but:
(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 contextually salient by being the meaning of some overt expression

Problems/questions/observations:

(a)   why should pronouns have these extra meanings/LFs? Are they ambiguous in the lexicon?
• free function variable approach: a pronoun can either be $\underline{x}$ or $\underline{f(x)}$ - just accidental homophony
• 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 being 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 $\underline{x}$
let it instead be a paycheck pronoun $\underline{f(y)}$ but where $\underline{f}$ is a function which maps each individual into $\underline{x}$ (i.e., into the individual assigned to $\underline{x}$ 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

Claims:
• 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

Step 1: • 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))

problems with this: (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)!!!

step 2: get rid of crucial use of the function variable (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   f[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' $--->_z$ hates; $(S/_LNP)/_RNP^{NP}$;   f[x[hates'(f(x))(x)]]

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

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

$=$ D[g[f[x[hates'(f(x))(x)]](D(g))]] $=$ D[g[x[hates'(D(g)(x))(x)]]]

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

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

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

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

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

V[f[V(f)(b)]]

Bill hates her; $S^{(NP^{NP})}$;   f[g[x[hates'(g(x))(x)]](f)(b)] $=$ f[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 _f[f]_ is just **g** applied to the ordinary meaning.
(143) **g**(her') = **g**(x[x]) =   f[y[x[x]](f(y)) =   f[y[f(y)]] =   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 it$_{r(k)(l)}$ hadn't been mailed yet

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

it' (in lexicon):   x[x] (type <e,e>)
        $-->_g$ f[f] (as given above) (type <<e,e>,<e,e>>)
        $-->_g$ (type < <e,<e,e>>, <e,<e,e>> >   V[x[f[f](V(x))]] (for V of type <e,<e,e>>)
                        $=$ V[x[V(x)]] $=$ 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.
                <u>Prediction</u>: 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
         no 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 <u>range of the function</u> which specifies the gender of the
         pronoun, 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
                individuals

(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
         range of x?

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

                           NP[+Fem]   (pick your favorite structure:
                                                              the idea is to have a structure
                    N[+Fem]   PP   analogous to *the mother of Bill)*

                     her   pro

                     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
         arguments 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   w[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 <u>range
of the function</u> - this is because the range is what is inherited from the original meaning

If gender is syntactic: no problem either

her; NP[F]$^{NP[F]}$;   w[w] --> **g** her; (NP[F]^{NP})$^{(NP[F]^{NP})}$;   F[x[w[w](F(x))]] (for F a
variable over functions of type <e,w>) =   F[x[F(x)]] =   F[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" by
the head.

<u>the mystery</u>: 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:
     x[wife'(x's childhood sweetheart)(x)]

in framework here: assume relational nouns <e,<e,t>>   --->**z** gives i-within-i violating
meaning:

(148)   wife-of' --->**z**   f[x[wife-of'(f(x))(x)]]
            her-childhood-sweetheart' = the-childhood-sweetheart-of'
            wife-of-her-childhood-sweetheart'   =   x[wife-of'(the-childhood-
            sweetheart-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
                participles, 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
            variable over categories)
   • this motivated by whole CG program (where syntax and semantics "mesh", and
            where syntactic combinatorics regulate semantic combinatorics)

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

--> hence; wife; N/PP[OF];   x[y[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
syntactically 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
        agentive 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 $\underline{f(x)}$ [x kissed the y: woman'(y) and y loves x]
>   (using Cooper/Engdahl)
>   b. the x: man'(x) and x loves $\underline{\text{the z : woman'(z) and z loves x}}$ [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
        representation
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):

  f[the man who **z**-loves f **z**-kissed the-wife-of function]
  combinatorics for (160a):

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

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

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

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

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

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

  =   f[the   x[wife-of'(f(x))(x)] **z**-kissed   y[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
      the 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 $((A/_LNP)/_R(VP/_RNP))/_RPP$

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 individual

It has associated with it some entailment such that for all individuals x and y and all relations 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 portion in (33) and (34) would appear to have exactly/essentially the same syntactic representation 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

Case 2: Parasitic gaps:

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 something 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 domain in the above, but also to the binder. If the binder is a "true quantificational" binder the constraint holds; if not the constraint doesn't hold.

preliminary note:

• <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 of 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 **his$_i$ mother liking __$_i$**

            --/--> (standard theory)   x's mother like x

--/--> (variable-free theory)   x[x's mother like x]   - this would be possible, but would require 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 semantic composition
                            --->   x[y[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 *easy*
• the subject position of *easy* moreover "controls" the object gap position, via lexical entailments

<u>Case II</u>:

38. Who$_i$ did you fire __$_i$ before **his$_i$ mother had a chance to warn __$_i$**?

            --/--> (standard theory)   x's mother had a chance to warn x

            --/--> (variable-free theory)   x[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":
                    --->   x[y[x's mother have a chance to warn y]]
• hence no Weak Crossover violation
• each gap bound separately, by the object position of *fire*

• the details:

(**a**) How can the object position of *fire* bind into the adjunct? If the adjunct is a VP modifier, it will be introduced only <u>after</u> the object is introduced; hence the object is lower and so this itself should introduce a $\overline{\text{WCO}}$ violation.

 <u>Assumption</u>:   The *before* - clause is actually introduced before the object. Assume that it can 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 alternatively 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 assumptions   have been made/argued for in a variety of frameworks - see, e.g., Pesetsky (1994) and Hukari 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 gap position:

fire-__-before-Bill-could-warn-__'   =   x[fire'(before-Bill-could-warn'(x))(x)]

Mary-fire-__-before-Bill-could-warn__' =   x[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-commanded by the other gap

something has to give with respect to this assumption or with respect to Weak Crossover 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 support
        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' =
  x[y[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, and "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 position to the object slot.

The pronoun and the gap are thus ultimately "merged", even though they are not "merged" in the meaning of *his mother had a chance to warn __.*

Antecedent Contained Deletion with Direct Compositionality

Background: VP Ellipsis in general

(1)   John will eat beans, and Bill will too.

most   analyses adopt some combination of the following two cross-cutting parameters:

(a)   deletion (under some kind of identity with some other expression, see below)   vs.
        interpretation (supply something)

(b)   identity of LF   vs.   identity of meaning

        Note: Tancredi (1992), Rooth (1994) and others: in addition to identity, there is
               also a focus condition:
               roughly: the ellipsis site must be contained in some constituent C such that
               the antecedent is contained in a constituent C'whose meaning (or something
               entailed or even implicated by the meaning of C') is an alternative to the
               meaning of C

        e.g. - *John will eat beans*   is an alternative to *Bill will (eat beans)*
               and *Bill*   must be focussed, hence "invoking" a set of alternatives

        Moreover: recent attempts to get rid of identity condition altogether in favor of
               some extension of the focus condition   (Tancredi, 1992, Fox, 1999,
               Merchant, 1999)

- Deletion + Identity of LF: Sag, 1976
- Interpretation + Identity of LF (= copy in an LF): Williams, 1977 and many others
- Deletion + Identity of Meaning: ??? (but compatible with approach here)
- Interpretation + Identity of Meaning (e.g., pick up a meaning): approach here (see also,
        Partee and Bach (1981), Hardt (1992), Dalyrmple, Shieber, and Perriera (1992),
        etc.)

Assumptions here (some made purely for expository convenience):
               (for more detailed treatment, see Jacobson, to appear in J. Kruiff and R.
               Oehrle (eds.), *Binding and Resource Sensitivity)*

- think of this like "free pronouns" - think of auxiliary as being like a pronoun
        (here, a pro-VP) (thus the term *VP Anaphora* is actually correct)
- for simplicity: we can think of this as encoding a "free variable"
- so *will* -   basic item is a VP modifier
               maps to second item includes a "free variable" - it is a VP proform
                     and means will'(P)

(1)   John will eat beans and Bill will too.
               Bill will too ;   b will'(P)
        value for P gets contextually specified (like free pronouns), where it is made
        contextually salient by meaning of *eat beans*

Two obvious things to say more about:

(i)   observations in Hankamer and Sag (1976):   unlike free pronouns, this is <u>not</u>   an
        instance of "deep anaphora" - the ellipsis site meaning <u>in general</u>   cannot be
        supplied purely by context, but must be supplied by overt linguistic material:

        <u>BUT</u>: Since Schachter (1977): we know that it in fact <u>can</u> be supplied by context,
        especially if the context is "rich enough"
                for various relevant examples: Schachter, 1977; Webber, 1978; Hardt,
                        1992; Dalyrmple, Shieber and Pereira (1992) among others

        <u>the hope here</u>: it is supplied by context, but the type of object that we need to
        supply (a pure function of type <e,t>) is very fragile and not easily recovered just
        from context
                • well known that it not only prefers a linguistic antecedent, but that it
                        prefers a recent one
                • hence, it very much likes to be supplied by being the meaning of some
                        linguistically overt expression
                • *thus: as is standard - this can be used (generally) as a diagnostic for the
                        meaning of the antecedent*

        <u>questions for further research</u>:

                • *why* should this kind of object be "fragile" and difficult to pick up just
                        from the context (or even from a meaning supplied in the non-recent
                        discourse)?
                • more specifically: this picks up something of type <e,t>
                        but anaphora with *it* in "do it" construction is picking up something
                        very close in meaning
                • speculation: there's a slight difference between functions of type <e,t>
                        and their "individual correlates" (cf., Chierchia, 1984) such that the
                        latter are more stable, the former are not (but why??)
                • note the similarity with the situation with paycheck pronouns

(ii)   use of free variable: this is a cheat
                • see Jacobson (to appear); this is just for expository convenience and is trivial to
                        fix
                • will; VP/$_R$VP; will' ---> (maps by a rule which applies to all aux's)
                                will; VP^{VP}; will'
                • hence, it's a kind of pro-form, and the information that anything containing it
                        contains something unbound within it will be "passed up", so that
                                Bill will

                        will mean: P[will'(P)(b)]]   (as in the case of any other "free" pronoun)
                • an interesting question is whether or not this is ever "bound" by the normal
                        binding processes; see various literature on this

3. <u>ACD: The received wisdom</u>

3.1. <u>Background: The received wisdom on relative clauses</u>

(2)   John will read every book which Mary will read.

axiomatic assumptions in lots of work in syntax and semantics

(i)   the only method of semantic combination is by functional application
an even stronger assumption which is very standard:
(ii)   all functions must be "saturated" - they must get all their arguments for semantic
          composition to proceed
• (i) follows from (ii); from (ii) also follows an additional thing (iii):
(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.)

assume: <u>will'</u> is a VP modifier (maps (time-dependent) sets   into (time-dependent) sets

              <u>will'(leave')</u> = a set of individuals x such that in the future leave'(x)

              <u>will'</u>   =   P[x[**F** (P(x))]]   (roughly)
              informally: give me a property, and I'll give you back the set of individuals
              who will have that property in the future

(3) semantic composition of *every book which Mary will read*

              <u>will'</u> needs to find a VP meaning
              <u>read'</u> is a 2-place relation among individuals (not a VP-type meaning)

              so: for the semantic composition to proceed: there must be a hidden object (a trace
              in the syntax and/or a level of representation in the syntax <u>and a variable in the</u>
              <u>semantics</u>)   to give a VP- type meaning

              read t   <u>read'(x)</u>
                      (read'(x) has as its meaning a property relative to ways to assign values to
                          x)
              will read t: <u>will'(read'(x))</u>
              Mary will read t: <u>will'(read'(x))(m)</u>   or   (for notational ease):
                                <u>m will(read'(x))</u>

              <u>m will'(read'(x)):</u> a proposition relative to ways to assign values to <u>x</u>

              this them mapped into a "closed property" by   -abstraction   --->

                       <u> x[m will (read'(x))]</u>
              = a "closed" property (on all ways to assign values to x, this is):
                      the property of being an x such that Mary will read x

              in set terms: {x]m will read x}

              book which Mary will read: intersect above set with <u>book'</u> set

3.2. <u>The consequences for ACD</u>

(4)   John will read every book which Mary will

         (a) as above: <u>will'</u> can only be happy if it finds a VP-type meaning (a set)
         (b) some kind of missing meaning is supplied (either deletion or interpretation)
         (c) missing meaning must be meaning of some (overt) linguistic expression

(identity condition)
(d) <u>ergo</u>: there has to be some linguistic expression at some level to supply a VP-
            type meaning

but: on surface: there is no VP which can supply the missing meaning
                    only candidate is *read every book which Mary will*
                        • but that can't be the one, since that leads to an infinite regress
                        • i.e., that leads to the "antededent containment" paradox

<u>LF to the rescue</u>:

    • posit a level of representation at which the object NP is "pulled out" of the clause,
    but where there is a trace/variable in object position   ("Binders Out" approach)
    e.g.,   Quantifier Lowering: McCawley (1967), Lakoff (1971)
            Quantifying-In: Montague (1973) (technically not quite the same as
                    positing LF, but does use a Binders Out level
            Quantifier Raising: May (1976)

(5)   for every book, x: [Mary will ___] , [John will **read x**]

3.3 <u>The fallacies here</u>:

• no reason to assume only functional application

OR: *You can't (don't) always get what you want*

• Steedman (1987), Dowty (1989) and many others: can put relative clause meanings
        together directly, and without traces and/or variables in object position, by allowing
        function composition
        • informally - an <u>incomplete</u> expression like *read* - which hasn't found it's object
                can directly combine with the auxiliary, and so on "up the tree"

(6)   John will read every book which Mary will read.
        let <u>read'</u>   function compose with <u>will'</u>
                <u>read'</u> is a function of type <e,<e,t>>
                <u>will'</u> is looking for an <e,t>, and thus is a function of type <<e,t>,<e,t>>

                will' $_o$ read' =   x[will'(read'(x))]

        this then function composes with the meaning of the subject

        hence: assume that <u>Mary'</u> is a function from VP meanings to S meanings

          P[P(m)] (i.e., <u>the set of properties that the individual m has</u>)
                Note: This assumption is independent of the domain here; it is Montague's
                treatment of NP meanings in general, set up to give a meaning to quantified
                NPs; justified also for coordination of quantified and ordinary NPs

        <u>Mary' $_o$ x[will'(read'(x))]</u> =   P[P(m)] $_o$ x[will'(read'(x))]   =
                 y[P[P(m)](x[will'(read'(x))](y)]] =

$\quad$ y[P[P(m)](will'(read'(y))]] $\;=\;$
$\quad$ y[will'(read'(y))(m)]

for notational simplicity: $\quad$ y[m will'(read'(y)]

• hence: final meaning same as in standard treatment
$\qquad$ • a set, which intersects with the head (ignore contribution of relative pronoun for
$\qquad\qquad$ now)
$\qquad$ • no need to posit a stage with an unbound variable in object position, which
$\qquad\qquad$ eventually gets bound

• note: have used a variable in the representation - but this treatment is *variable-free*
$\qquad$ • nothing ever has as its meaning a function which varies according to how we
$\qquad$ assign values to the variables
$\qquad$ • so we can strip away the assignment functions altogether

3.4. <u>The consequences for the analysis of ACD</u>

• Direct compositionality account of ACD (Cormack, 1984; Evans, 1989; Jacobson, 1992a,
$\qquad$ 1992b):

• the idea in a nutshell: the "missing meaning" here is just the 2-place relation <u>read'</u>
$\qquad$ (supplied by the meaning of the overt expression *read)*   - and it combines up with
$\qquad$ the auxiliary just as it does in the full case

<u>in more detail</u>:

recall *will* as an anaphor: $=$ <u>will'(P)</u>
$\qquad$ (more technically, its meaning is   P[will'(P)] which $=$ will'
assume also: <u>will'$_{\mathrm{o}}$ R</u>
$\qquad\qquad$ (that is, it is looking for a 2-place relation to compose with)
$\qquad$ (more technically, its meaning will be: R[will' $_{\mathrm{o}}$ R]

NOTE • This looks like a kludge - why the second *will?*
$\qquad$ • the beauty of the standard analysis is that ACD $=$ VPE, and so nothing extra
$\qquad\qquad$ needs to be said
BUT: • once the full details of the system are supplied the same is true here; this
$\qquad$ second*will*   (the one occuring in the ACD case) is automatic; nothing extra is
$\qquad$ needed

$\qquad$ • most general form of the **g** ("Geach") rule will map:
$\qquad\qquad$ A$^{\mathrm{B}}$ to (A/C)$^{\mathrm{B/C}}$
$\qquad$ • i.e., I contain a pro-form which has a "gap" of type C, and - modulo that - I am
$\qquad$ something with a gap of the same type

$\qquad$ • hence: will;   VP$^{\mathrm{VP}}$;   will'$\qquad$ --->$_{\mathbf{g}}$ $\qquad$ will;   (VP/$_{\mathrm{R}}$NP)$^{\mathrm{(VP/_RNP)}}$;
$\qquad$ R[x[will'(R(x))]]   which is equivalent to: R[will' $_{\mathrm{o}}$ R]

$\qquad\qquad$ in other words, this is nothing more than a pro-form over transitive verbs,
$\qquad$ rather a VP proform

everything else proceeds in the usual way

• but, for expository simplicity, will think of *will* as looking for a 2-place relation to function compose with

(7)   John will read everything which Mary will.

will' ₀ read' proceeds exactly as in (6)

again: in reality <u>will'</u> itself doesn't "pick up" <u>read'</u> - it just passes up the information that there is a missing TVP meaning, so that the final thing will end up being:
<u>R[j will read everything which Mary will R]</u>
and this is applied to <u>read'</u>

a related but different proposal: Lappin, 1995 - ACD as pseudo-gapping

3.5. <u>Interlude</u>:
Standard view: ACD is a special case of VP Ellipsis
this view adopted here
Is TV(P) Ellipsis really same as VP Ellipsis?

• as with VP Ellipsis, it can happen across Ss in a discourse:

(8)   a. VP Ellipsis: I ran. Bill did too.
b. TVP Ellipsis: Bagels, I like. Donuts, I don't. (Evans, 1989)

NOTE: This is exactly the kind of case that Sag's condition on alphabetic variance was designed to preclude. However:

(i) his observations about the data were incorrect. One sort of case was:

(8c)   Which book does Mary like? I don't know, which book does Sue?

Evans (1989) observes that this probably has something to do with the subject/aux inversion, note:

(8d)   I know which book Mary likes, and I know which book Sue doesn't.

Additional cases which show that Sag's generalization was mistaken:

(8e)   John$_i$ asked Mary to water his$_i$ plants, and Bill asked Sue to. (Jacobson, 1992; see also Hardt, 1992 for related cases) (sloppy reading okay)

(ii) Sag's definition of alphabetic variance was non-standard:
• Given two identical formulas F_1 and F_2 each of which contains within them an unbound occurence of <u>x</u>, they are alphabetic variants only if the two occurrences of <u>x</u> are ultimately (higher up) bound by the same binder
Example: the elided VP and its antecedent in (8e) can both be represented as:

x[x water y's plants]
but the condition was defined to keep these from being alphabetic variants, since the two occurrences of <u>y</u> are separately bound

• usual definition of alphabetic variance: two formulas are alphabetic variants iff they <u>mean</u> the same thing; here the two formulas do mean the same thing, and yet they are defined as non alphabetic variants - hence Sag's condition can't be stated compositionally (see Partee and Bach, 1981 for discussion) - have to look higher in the tree; and is non-standard

hence, it is a happy fact for everyone that the initial generalization was incorrect

• as with VP Ellipsis, ellipsis site can precede antecedent:

(9)   a. VP Ellipis case: Before John did, Bill climbed the mountain.
        b. TVP Ellipis: ? Everything which Bill does, I'll also read.

• as with VP Ellipsis, with enough work: can have a contextually salient but not linguistically overt antecedent:

(10)   a. VP Ellipsis case: (i)   I see you making salad, I pick up some carrots and a knife and say: Do you want me to?   or   Should I.
        b. TVP Ellipis case: Context: I see you about to reach for some cookies that just came out of the oven. Pointing first to a batch on a different plate, and then to the hot batch:

        These, you may. Those you mustn't, at least not until they cool down.

• A question (for both the standard view and the view here):
        • all of these are harder to construct with TVP Ellipsis (or, in standard theory, with a VP containing a variable in object position) - *why?*
        • tentative answer here: 2-place relations are even more "fragile" and difficult to access from context or from distant linguistic material than are meanings of type <e,t>

3.5.   <u>TV**P(!)** ellipsis - not just simple transitive verb</u>

(11)   John thought that Bill would read everything that Sally also did *(think that Bill would read)*

• analysis has been developed to handle scope interaction facts in Sag (1976) - see Cormack (1984) and Jacobson (1992a)
• and various facts in Larson and May (1989), see Jacobson (1992a)
• plus additional interactions with Heavy NP Shift (Jacobson, 1992a)

 4. <u>But wait:</u>

(12)   John voted for every girl who thought he would. (*vote-for her*)

the apparent problem for the ACD as TVP Ellipsis analysis:

• what's missing here is a VP with a pronoun/variable in object position
                vote-for'(her')

(13)   a. John voted for every girl who thought he would vote for her.
         b. *John voted for every girl who thought he would vote for.

• hence: need to supply vote-for'(x)
• in that case, we are back to the original argument for Binders Out level where the object
         is pulled out
• there is no surface VP whose meaning is vote-for'(x)
• but there is such a VP if we pull out the object

the fallacy: thinking that to do the semantic compositon we need a *variable* in object
         position

5. But: Variable-Free semantics

 Implications for the ACD case

(28)   John voted for every girl who thought (that) he would.

Interlude: dealing with the pronoun he

Note: these best with a pronoun in subject position (Haik, 1985)

(29)   *John voted for every woman who thought (that) Bill would.

• claim: this has to do with focus and stress (not with the fact that we can have a bound
         pronoun):

(30)   ?John voted for every woman who thought (that) John would. (at worst, a mild
         "Principle C" violation)

(31)   The Dating Game scenario: Imagine a t.v. show in which two men and two
         women are contestants each evening. Each woman writes down the name of one
         man that she would like to date. Each man writes down the name of a woman
         who he thinks will pick him. If the woman A picks the man B and B succeeds in
         guessing that A will pick B, then A and B get to go out on a fabulous date.
         But last night, no one one, because

         a. Unfortunately, Mary picked the man who thought that Sue would pick him, and
         Sue picked the man who thought that Mary would.
         b. ?Unfortunately, Mary picked the man who thought that Sue would, and Sue
         picked the man who thought that Mary would.

to ignore irrelevant complication, will deal with (30)

(30)   John voted for every woman who thought that John would.

the solution:
         • *vote-for her* means the same thing as *vote-for*
         • i.e., only need to pick up the TVP meaning vote-for'

- the "object position" of this 2-place relation is bound in the same way that variable-binding happens in general
- in this case: *thought*   undergoes **z** - so its subject slot "binds" the object position of *vote-for*

(31)   *would;* would' $_o$ R
        R can be "filled in" as vote-for'

        would' $_o$ vote-for' =   x[would'(vote-for'(x))]
        John - type lifted meaning, composes with this --->
                        x[j would'(vote-for-(x))]   (type <e,t>)
        thought; **z**(thought') =   P[y[y thought P(y)]]

        thought John would vote for;   P[y[y thought P(y)]](x[j would'(vote-for-(x))])
                =   y[y thought (j would vote-for (y))]

        this then intersects with woman'   (set denoted by head)

(32)   *John voted for every woman who thought John would vote for
                badness from the syntax, not the semantics
                        (for details, see Jacobson, to appear)
                there is no way for the syntax to allow this - the story is exactly analogous to
        any version of a standard story
                        details (both in standard story and here) depend on precise syntax of
                        subject *who* - assume it's an ordinary subject (not "extracted") so it
                        wants to combine with a VP
                                i.e.: RC/$_R$ VP
                - then it cannot combine with a transitive verb
                - it can combine with a VP with a pro-form, by **g**   - but that's not
                what it finds here
                - it can combine with a VP with an extraction gap, but then that will
                be passed up and eventually has to be in a construction licensing that

                        *further footnote: one does ultimately need to say what are the
                        set of "standalone" categories*

a remark: we would expect this to show the same properties as other cases of (T)VP
        "ellipsis":

-- across sentences in a discourse:

(33)   John spoke to several girls. But Mary is the only one who wanted him to.

-- very difficult to get a "deep anaphora" case, but marginally possible with work:

(34)   Context: (with apologies to the LSA regulations): a party with a bunch of junior high
        school students, and Johnny is going around trying to kiss every girl. I say to him:
        Hey, I think Sally is the only one who wants you to.

6. An advantage over the stnadard approach: Pied Piping and Ellipsis

6.1. Pied-piping without reconstruction

- the semantics of Pied-Piping is now automatic, making only one pretty reasonable assumption) without any kind of reconstruction
   caveat: -- this is true for relative clauses
              -- obviously, this needs to also work for questions
              -- whether or not the extension to questions is automatic remains to be seen

(33)   every man the father of whom Mary voted-for

- in functional questions, *vote-for* can undergo **z** to want a functional gap
              (whose argument is bound by the subject position of *vote for)*
- but it can also undergo **g** to want a functional gap
          whose argument position is not bound (yet), and is "passed up"

- another (simpler) derivation which gets the same result:
      let ordinary *vote for* function compose with *Mary* (in the normal way)

$$\lambda P[P(m)] \ o \ \underline{vote\text{-}for'} \ = \ \lambda x[\ m \ vote\text{-}for \ x]$$

          i.e., an <e,t> - or, a set
      but this can undergo **g** so that it wants not an e-argument, but an <e,e> argument
          i.e., it characterizes a <u>set of functions of type <e,e></u> (rather than a set of individuals

$$\mathbf{g}(\ \lambda x[\ m \ vote\text{-}for \ x]) \ = \ \lambda f[\ \lambda y[\ \lambda x[m \ vote\text{-}for \ x](f(y))]] \ =$$
$$\lambda f[\ \lambda y[m \ vote\text{-}for \ f(y)]]$$

          informally: give me a function f (from individuals to individuals) and I'll give you the set of individuals y such that mary votes for f(y)

      <u>so, this wants a function as argument</u>

   the father of whom:
          - assume that relative pronouns have the semantics of ordinary pronouns
          - then, *the father of whom* is just the-father-of function
          - and so is of the right type to be argument here

              λf[λy[m vote for f(y))]](the-father-of)   =
              λy[m vote for the-father-of(y)]

A note of "semantic reconstruction": have shown out the -conversion step above, and in some literature this gets referred to as "semantic reconstruction"

      •   -conversion is only a notational device; the two formulas above ar ethe same model-theoretic object; it's just two different ways to show and to notate the same thing
      • hence no real sence in which the-father-of function is being "put back" into some syntactic representation

this now denotes a set, and so can intersect with the head

Note: a similar analysis proposed within the standard theory in Sharvit, 1998
      Sharvit also points out that given functional questions, can have a functional gap here

but here, both parts come for free:
> • the possibility of a functional gap comes from **g** which is part and parcel
of the mechanisms needed for variable-binding anyway
> • the fact that *the father of whom* denotes a function is automatic from the
semantics of pronouns

6.2. <u>The payoff: the interaction of Pied-Piping with ACD</u>

NOTE: <u>assume</u> the traditional view: there is an identity condition (on meaning and/or LF)
>with the antecedent

>will return (if time) to consequences for views which dump this condition

(34)   Mary voted for every candidate the FATHER of whom Bill had.

the problem: how can their be identity between the meaning (or, LF) of the matrix VP and
>the meaning (or, LF) of the ellipsis site?

(i) assume a reconstruction analysis of Pied-Piping
> • then the meaning of the matrix VP (assuming Binders Out) is:
>> <u>vote-for'(x)</u>
> • the meaning of the ellided VP is <u>vote-for'(the-father-of'(x))</u>

(ii) assume something like Sharvit's analysis - where the gap is functional
> • then, the meaning of the matrix VP (assuming Binders Out) is:
>> <u>vote-for'(x)</u>
> • the meaning of the elided VP is <u>vote-for'(f(x))</u>

note: much discussion in traditional view about alphabetic variants
>one view: identity allows identity up to difference in variable names:

>one version:
>> vote-for'(x)   =   vote-for'(y)
>another version:
>> $x[x$ vote-for'$(x)]$ = $y[y$ vote-for'$(y)]$

>the ned to define "identical up to alphabetic variance" does not arise in Variable-Free
>> semantics
>variable names are an artefact (just convenient notational labels)
>the fact that the above two formulas are equivalent is automatic, since they just
>> name (the same) model-theoretic objects

but in any case: <u>f(x)</u>   <u>x</u>   and so <u>vote-for'(f(x))</u>   <u>vote-for'(x)</u>

why the analogous problem does not arise in the account here:
> • <u>vote-for'</u> is the meaning that we are picking up
> • i.e., the TVP ellipsis analysis says identity is just identity of TVP meaning

a potential worry: Are we equating **g**(<u>vote-for'</u>) with <u>vote-for'</u>? and hence doing the same
>kind of cheat?

Answer: no. We can just "pick up" <u>vote-for'</u>. The presence of the missing functional
    argument (which is filled by the meaning of *the father of whom)* comes about from
    g on the whole thing - <u>not</u> **g** on the "missing meaning"

(34)   Mary voted for every candidate the FATHER of whom Bill had

    had; <u>had'$_o$ R</u> = <u>x[had'(R(x))]</u>

    pick up: <u>vote-for'</u> as value for R, so: <u>x[had'(vote-for'(x))]</u>
           NOTE: there really is no reason to think that this happens in the
    compositional semantics in this way
           • in reality, the whole thing is a function from 2-place relations to the rest,
    and the value is supplied "at the end"
           • however, the exposition is much simpler if we supply it right away

    Bill had;   <u>P[P(b)]</u> <u>o</u> <u>x[had'(vote-for(x))]</u> =

        <u>x[P[P(b)](had'(vote-for(x))]</u> =

        <u>x[had'(vote-for'(x))(b)]</u>

         notational ease: <u>x[b had'(voted-for'(x))]</u>

    ---> **g**:

        <u>f[y[x[b had'(voted-for'(x))](f(y))]]</u> =

        <u>f[y[b had-voted-for'(f(y))]]</u>

             i.e., give me a function and I'll give you back the set of y such that
                b voted for f(y)

         this then gets *the-father-of* function and returns the set of y's such that b
           had voted for the father of y

crucially: we didn't apply **g** to the missing meaning, so we didn't skirt the identity
    condition

this works, because we don't need to pick up a full VP meaning, complete with a
    functional gap and an argument position for that gap (where there is no other VP to
    supply this meaning)
    we need only pick up a TVP meaning; the functional argument position is supplied
    by the operation of **g** (which is part and parcel of the whole binding system)

6.3. <u>is there a problem in the classical (VP Ellipsis) account of the identity condition is</u>
    <u>dropped</u>?

    • depends on what replaces it

Rooth's focus condition by itself is not enough - since deaccenting and ellipsis not the
    same:
(35)   Mary called Bill an idiot, and then SALLY insulted him too.
          (small font indicates de-accenting)

(36)   Mary called Bill an idiot, and then SALLY did too.
                does not mean: Sally insulted him too.

NOTE the following asymmetry:

(37)   ??Every candidate the FATHER of whom BILL had, MARY voted for.
         marginal, because ellipsis site not like to precede antecedent, but not awful

(38)   *Every candidate the FATHER of whom BILL had voted for, MARY had.
         impossible on the relevant reading

compare to:

(39)   ??Every candidate who BILL had, MARY voted for. (also funny)
(40)   Every candidate who BILL had voted for, MARY did. (fine)

explanation in Jacobson (1998): asymmetry from Rooth's focus condition:

alternatives to *the FATHER of whom BIll had* in (37):
   • sets of individuals of the form   y[a voted for f(y)]
           which vary on the value of a (the voter) and f (the function)

meaning of *MARY had (voted for)* (after supplying the "missing meaning") =
   •   y[m voted for y]

hence, doesn't directly satisfy focus condition
<u>but</u>: implicational bridging: from this can derive
   •   y[m voted for id(y)]

       that is: identity function can be the alternative here to the FATHER function above
       • for independent evidence that identity function is a real alternative to more run-of-
       the-mill functions, see Jacobson (1998)

reverse does not work:

we can't get the meaning of *the FATHER of whom BILL had voted for*   to be an
       alternative to
       MARY had (voted for)
because there is no overt material to allow <u>id</u> to be in focus (and hence no overt material to
       send us off looking for an alternative to the identity function)