**On Apparent Exceptions to Weak Crossover in a Variable-Free Semantics**

Pauline Jacobson
Dept. of Cognitive and Linguistic Sciences, Box 1978
Brown University
Providence, RI 02912 USA
pauline_jacobson@brown.edu

Conference on Non-Lexical Semantics
Universite de Paris VII
June 21, 1996

## I. Overview

• apparent exceptions to Weak Crossover (Lasnik and Stowell, 1991)

• approach to semantics making no essential use of variables (or indexing)

• a seemingly odd property of variable-free semantics, centering on the treatment of two pronouns and / or gaps which are bound by the same thing

• the apparent exceptions to Weak Crossover become unproblematic given this treatment

## II. Apparent Exceptions to Weak Crossover

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:

1. 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)

2. 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):

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

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

4. 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:

5. 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:

6. 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

7. 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

8. John is easy for me [$_{CP}$ $wh_i$ [PRO to please $\_\__i$]]

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

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

within Categorial Grammar (see Jacobson, 1992):

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

9. 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:

10. 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)

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

12. 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 __$_i$?

13. *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 (14) and (15) would appear to have exactly / essentially the same syntactic representation and the same meaning:

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

15. 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:

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

17. *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 (17) from translating as:

**x's mother had a chance to warn x**

The same principle should keep the boldface domain in (16) 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.

## III. The intuition behind the solution here

• In the relevant cases, the boldface portion does not translate as **x's mother like/warn x -** nor as anything analogous to this

• Using variables for the sake of illustration, it would be as if the portion translates as:
**x's mother like/warn y**

• The two positions (the pronoun and the gap) nonetheless end up being "merged" later in the semantic composition - they are, in essence, "cobound" in a way which will be made precise

• A standard semantics with variables could conceivably make use of this same solution, depending on what other assumptions are made. However, this solution is completely natural in a variable-free semantics - in fact, things could not be otherwise.

## IV. Variable-Free Semantics

(Curry and Feys, 1958; Quine, 1966; more immediately, Szabolcsi, 1987, 1992; Hepple, 1990; most immediately, Jacobson, 1991, 1992, 1994, etc.)

18. Every $man_i$ thinks (that) $he_i$ lost.

• Standard Account:

*he* --> x
*he lost* --> lost'(x)
*thinks (that) he lost* --> think'(lost'(x))

lost'(x) is an "open proposition" - i.e., a proposition relative to ways to assign values to variables -
hence: it is a function from assignment functions to propositions
hence: assignment functions are model-theoretic objects
hence: variables are model-theoretic objects
(see also Landman and Moerdijk, 1983)

• Variable-Free account:

No variables (or assignment functions) as model-theoretic objects or as any part of the semantic machinery

*he lost* --> lost' (i.e., a function from individuals to propositions)

In general: any constituent C of category A which contains a pronoun which is unbound within C denotes a function from individuals to whatever type of meaning constituents of category A normally have (and its syntactic category is actually $A^{NP}$ )

If a constituent C of category A contains two pronouns unbound within C, then it denotes a function of type <e,<e,A'>> (and its syntactic category is actually $(A^{NP})^{NP}$ )

19. Every $man_i$ loves $his_i$ mother.

• Standard account: *his mother* --> x's mother
• Variable-free account: *his mother* --> λx[the-mother-of'(x)] = the-mother-of' function (function of type <e,e>)

NOTE: Variables used in the representations of the meanings, but this for convenience only! No expression has a meaning which is a function from assignment functions (nor are meanings assigned relative to assignment functions).

• Binding under Variable-Free Account:

Take a verb (or other function) wanting two arguments, one of type Y and a later (higher) individual argument. It type shifts by a rule ***z*** such that instead of a Y argument, it wants an argument of type <e,Y>. Moreover, the newly created "open" e slot is "bound" to - or "merged" with - the higher individual argument slot of the verb.

In terms of the syntax: take a verb which wants as argument a constituent of category Y and a higher NP constituent. It shifts so as to take an $Y^{NP}$ instead of just a Y - i.e., it takes as argument a constituent containing a pronoun. (The semantics is such that the pronoun is "bound" to the higher NP argument slot.)

20. Let α be an expression of with meaning of type <Y',<e,X'>>
and with syntactic category (X/NP)/Y.
Then there is a homophonous expression β with meaning of type <<e,Y'>,<e,X'>>
and of category (X/NP)/$Y^{NP}$
where β' = ***z***(α').

***z*** is defined as follows:

$$\boldsymbol{z}(f) = \lambda g[\lambda x[f(g(x))(x)]]$$

21. Every $man_i$ loves $his_i$ mother.

*love'* is an ordinary 2-place relation between individuals
***z****(love')* is a relation between individuals and functions of type <e,e>, such that to z-love a function f is to be an x who loves f(x).
*his mother* is the-mother-of function

Hence to z-love the mother-of-function is to be an x who loves x's mother.

22. Every $man_i$ thinks that $he_i$ lost.

*think'* is a relation between individuals and propositions
***z****(think')* is a relation between individuals and properties, such that to z-think P is to be an x who thinks P(x)
*he lost* denotes the property lost'

Hence to z-think the lost' property is to be an x who thinks x lost.

The ***z*** rule as given above binds a pronoun within a complement C to the next highest argument after C is introduced. It can straightforwardly be generalized to allow binding to any higher argument slot (see Jacobson, 1992).

• A variety of arguments for this approach given in various papers of mine, including SALT 2,4, and 6, and Amsterdam Colloquium 8 and 9

• Binding into adjuncts: (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? For expository purposes, assume either free or a limited amount of 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*.

• Weak Crossover Effects:

***z*** is the only binding rule. ***z*** has the effect of binding a pronoun within a constituent C to an NP argument slot which is a higher argument than C

hypothesis: there is no "backwards" binding rule ***s*** which does just the opposite (***s*** has the effect of binding a pronoun within a constituent C to an NP argument slot which is a lower argument than C)

23. The ***s*** rule:

Let $\alpha$ be an expression with meaning of type <e,<Y',X'>>
and with syntactic category (X/Y)/NP
Then there is a homophonous expression $\beta$ with meaning of type <e,<<e,Y'>,X'>>
and with syntactic category $(X/Y^{NP})/NP$
where $\beta' = \boldsymbol{s}(\alpha')$.

***s*** is defined as follows:
$\boldsymbol{s}(f) = \lambda x[\lambda g[f(x)(g(x))]]$

24. *$Who_i$ does $his_i$ mother love $\_\__i$?

would be possible with ***s***, where the object "slot" binds the pronoun in the subject

loves; $(S/_L NP)/_R NP$; loves'
$\boldsymbol{s}$(loves); $(S/_L NP^{NP})/_R NP$; $\lambda x[\lambda f[loves'(x)(f(x))]]$

his mother; $NP^{NP}$; the-mother-of' --> **lift** ( the same way that any subject can type-lift):
$S/_R(S/_L NP^{NP})$; $\lambda W[W(\text{the-mother-of}')]$ (for W a variable of type
<<e,e>,t> - i.e., it's a variable over sets of functions of type <e,e>

**lift**(his mother) $\circ$ $\boldsymbol{s}$(loves); $S/_R NP$; $\lambda W[W(\text{the-mother-of}')] \circ \lambda x[\lambda f[loves'(x)(f(x))]] =$
$\lambda x[\lambda W[W(\text{the-mother-of}')](\lambda f[loves'(x)(f(x))])] =$
$\lambda x[\lambda f[loves'(x)(f(x))](\text{the-mother-of}')] =$
$\lambda x[love'(x)(\text{the-mother-of}'(x))]$

this then occurs as argument of the question pronoun *who'* - exactly as any other property can occur as argument of *who'* (and the syntax is the same as a normal constituent with an object gap)

FOOTNOTE: The above account has to be taken as a first approximation. There are well-known problems with the c-command condition on binding as an account of Weak Crossover (Reinhart, 1983) and most of these problems are inherited here. However, whatever fine-tuning is required, the point is that Weak Crossover effects can be accounted for without the use of variables, indices, etc. - *it is a constraint on the combinatorics which allow for the binding effect*

• Key Features:
- "binding" is a "merging" via ***z*** of the open argument slot in some constituent C which (syntactically) is occupied by a pronoun in C and some argument position which is higher than C
- hence, "binding" is to argument positions only (more specifically, to higher argument positions)
- "binding" of a pronoun within an adjunct is possible - for convenience we assume here type-lifting so that the adjunct can become an argument

**V. An apparent problem with not having variables**

• <u>How do binding when there is more than one pronoun and / or gap which "correspond to the same variable" (and where neither can bind the other)?</u>

25. Every $man_i$ thinks that **the woman who loves $him_i$ hates the dog that $he_i$ owns**.

26. the woman who loves $him_i$ hates the dog that $he_i$ owns

<u>Apparent advantage of variables: translate the two as the same variable</u>!!

**--->** the woman who loves x hates the dog that x owns

<u>Variable-Free account: no analogous way to do this</u>!

**--->** λx[λy[the woman who loves x hates the dog that y owns]]
(syntactic category is $(S^{NP})^{NP}$

<u>But</u>: <u>"Co-binding" is possible, and occurs via two applications of ***z*** on *think*</u>

27. /thinks/; $(S/_L NP)/_R S$; thinks'; ==>***z*** (shift by ***z*** rule):

/thinks/; $(S/_L NP)/_R S^{NP}$; ***z***(thinks') = λP[λx[thinks'(P(x))(x)]]
(note: thinks' is of type <t,<e,t>>,
so ***z***(thinks') is of type <<e,t>,<e,t>>)

==>***z*** (this then shifts again by ***z*** rule):

/thinks/; $(S/_L NP)/_R (S^{NP})^{NP}$; ***z***(***z***(thinks')) of type <<e,<e,t>>,<e,t>>

Thus, ***z***(***z***(thinks')) wants a 2-place relation R, and it is a relation between individuals x and 2-placerelations R such that x stands in the ***z***(***z***(thinks')) relation to R just in case x stands in the ***z***(thinks') relation to R(x). Hence; x stands in the ***z***(***z***(thinks')) relation to R just in case x stands in the ordinary think' relation to R(x)(x)

***z***(***z***(thinks')) = λR[λy[***z***(thinks'(R(y))(y)]] =
λR[λy[λP[λx[thinks'(P(x))(x)]](R(y))(y)]] =
λR[λy[λx[thinks'(R(y)(x))(x)](y)]] =
λR[λy[thinks'(R(y)(y))(y)]]

thinks-the-woman-who-loves-him-hates-the-dog-that-he-owns:

***z***(***z***(thinks'))(λx[λz[hates'(the-dog-that-z-owns')(the-woman-who-loves-x')]]) =
λR[λy[thinks'(R(y)(y))(y)]] (λx[λz[hates'(the-dog-that-z-owns')(the-woman-who-loves-x')]]) =
λy[thinks'(λx[λz[hates'(the-dog-that-z-owns')(the-woman-who-loves-x')]](y)(y))(y)] =
λy[thinks'(λz[hates'(the-dog-that-z-owns')(the-woman-who-loves-y')](y))(y)] =
λy[thinks'(hates'(the-dog-that-y-owns')(the-woman-who-loves-y'))(y)]

This then occurs as argument of the subject *every man*

• Difference between this and the standard account:

• In the standard account, where the two pronouns translate as the same variable, they are "merged" at the level of the meaning of the embedded S *the woman who loves him hates the dog that he owns*

• In the variable-free account, the two pronouns are not "merged" at the level of the meaning of the embedded S, but only later in the semantic composition (when this 2-place relation is taken as the complement of ***z***(***z***(think'))

FOOTNOTE: Depending on exactly how one implements the "standard" theory (i.e., a theory with variables), it can also allow for a derivation of this meaning which is analogous to the derivation above (in addition to the derivation where the two pronouns translate as the same variable). Whether or not there is such a derivation depends on just what other assumptions are made.

## VI. The apparent exceptions to Weak Crossover

Case I:

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

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

--/--> (variable-free theory) λx[x's mother like x] - this would be possible, but would require the use of ***s***(likes') - as in (24) above

• the basic intuition:
- the two do not "correspond to the same variable"; they are merged only later in the semantic composition
  ---> λx[λy[x's mother likes y]]
- since the pronoun is not "bound" to the object slot, there is no Weak Crossover violation
- the pronoun is "bound" by the subject position of *easy* - via an application of ***z*** on *easy*
- the subject position of *easy* moreover "controls" the object gap position, via lexical entailments

Case II:

29. 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) λx[x's mother had a chance to warn x] - this would be possible, but would require use of ***s***(warn')

• the basic intuition:
- again, the two do not "correspond to the same variable":
  ---> λx[λy[x's mother have a chance to warn y]]
- hence no Weak Crossover violation
- each gap bound separately, by the object position of *fire*