VFS Alternations

Phillip Potamites

April 17, 2007

1 Summary of P. Jacobson

(1) The Geach Rule: Passes a variable up 'over'.

a. $g_c(f)=\lambda P[\lambda x[f(P(x))]]$
b. $<a,b>$, $<<c,a>,<c,b>>$
c. $[\![$ love_g $]\!]([\![$ her $]\!])=$
d. $\lambda P[\lambda x[\text{love}(P(x))]](\lambda y[y])$[1]$=$
e. $\lambda x[\text{love}(x)]$[2]

(2) Jacobson's binding rule: Identifies the next arg up as the embeddee's desired arg. (see Jacobson (1999b), p. 134, ex. 23, for the properly generalized version; here we have the introductory version).

a. $z_b(f)=\lambda P[\lambda x[f(P(x))(x)]]$
b. $<a,<e,b>>$, $<<e,a>,<e,b>>$
c. $[\![$ love_z $]\!]([\![$ herself $]\!])=$
d. $\lambda P[\lambda x[\text{love}(P(x))(x)]](\lambda y[y])=$
e. $\lambda x[\text{love}(x)(x)]$

(3) Lift: Are all subjects lifted? g requires a lifted subject.

a. $l_b(a)=\lambda F[F(a)]$
b. a, $<<a,b>,b>$
c. $[\![$ Jill_l $]\!]([\![$ smokes $]\!])=$
d. $\lambda F[F(\text{Jill})](\lambda x[\text{x smokes}])$

[1] Following Jacobson, we here treat pronouns as a basic identity function from individuals to individuals, and abstract from gender, number, etc.. Note that $[\lambda y[y]](\text{Jill})=\text{Jill}$.

[2] Jacobson adopts this kind of combinatory semantics without much discussion of the implication on the configurational-lexicalist debates. Recall that, for her, $\lambda x[\text{love}(x)]=\text{love}$. Remember that I usually put more informative value in the presence or absence of these operators.

2 Some Alternations

2.1 Control

(4) [3]

a. Jill expected Jack to win.

b. $[\![$ expected$_1$ $]\!]=\lambda P_{<i,<w,t>>}\lambda x\lambda w$. expect(e) $\wedge$ t(e)<now(w) $\wedge$ ex(e,x) $\wedge$ pat(e,w') $\wedge$ P($\lambda e'$. t(e)$\leq$t(e'))(w')

c. Jill expected to win.

d. $[\![$ expected$_2$ $]\!]=\lambda P_{<e,<i,<w,t>>>}\lambda x\lambda w$. expect(e) $\wedge$ t(e)<now(w) $\wedge$ ex(e,x) $\wedge$ pat(e,w') $\wedge$ P(x)($\lambda e'$. t(e)$\leq$t(e'))(w')

e. $[\![$ z(expected$_1$) $]\!]=\lambda P_{<e,<i,<w,t>>>}\lambda x\lambda w$. expect(e) $\wedge$ t(e)<now(w) $\wedge$ ex(e,x) $\wedge$ pat(e,w') $\wedge$ P(x)($\lambda e'$. t(e)$\leq$t(e'))(w')

(5) shows the work a little more explicitly:

(5) z(expect):

a. $\lambda P[\lambda Q[\lambda y[$P(Q(y))(y)]]]($\lambda R_{<i,<w,t>>}\lambda x\lambda w$. expect(e) $\wedge$ t(e)<now(w) $\wedge$ ex(e,x) $\wedge$ pat(e,w') $\wedge$ R($\lambda e'$. t(e)$\leq$t(e'))(w'))=

b. $\lambda Q[\lambda y[\lambda w$. expect(e) $\wedge$ t(e)<now(w) $\wedge$ ex(e,y) $\wedge$ pat(e,w') $\wedge$ Q(y)($\lambda e'$. t(e)$\leq$t(e'))(w'))]]

For convenience, I repeat examples of the kinds of complements each function could, respectively, take as an argument.

(6) a. $[\![$ Jack to win $]\!]=\lambda i\lambda w$. win(e',w) $\wedge$ i(e') $\wedge$ ag(e',Jack)

b. $[\![$ to win $]\!]=\lambda x\lambda i\lambda w$. win(e',w) $\wedge$ i(e') $\wedge$ ag(e',x)

2.2 Intrinsic Reflexivity

(7) (representing just the relevant elements)

a. Jill shaved Jack.

b. $[\![$ shave $]\!]=\lambda x\lambda y$. shave(e,w) $\wedge$ ag(e,y) $\wedge$ pat(e,x), [acc]

c. Jack shaved.

d. $[\![$ shave $]\!]=\lambda y$. shave(e,w) $\wedge$ ag(e,y) $\wedge$ pat(e,y)

[3] Notice that I am applying this operation in the syntax, after tense has been assigned, so that I can claim that it is type-driven. This might be a little goofy, and it may, ultimately make more sense to consider it a lexical operation. The formulas would only require minor shifts.

2.3 Raising

(8) a. It was likely for Jill to win.

b. $[\![$ likely $]\!]=\lambda P \lambda i \lambda w$. likely(e,w) $\wedge$ pat(e,w') $\wedge$ i(e) $\wedge$ P($\lambda e'$. t(e)$\leq$t(e'))(w')[4]

c. Jill was likely to win.

d. $[\![$ g(likely) $]\!]=\lambda P \lambda x \lambda i \lambda w$. likely(e,w) $\wedge$ pat(e,w') $\wedge$ i(e) $\wedge$ P(x)($\lambda e'$. t(e)$\leq$t(e'))(w')

Notice that to use (d) we'll then need to lift and g 'was', because of the type mismatch with 'x'.

(9) a. $[\![$ g(likely) to win $]\!]=\lambda x \lambda i \lambda w$. likely(e,w) $\wedge$ pat(e,w') $\wedge$ i(e) $\wedge$ win(e',w') $\wedge$ ag(e',x) $\wedge$ t(e)$\leq$t(e')

b. $[\![$ was $]\!]=\lambda e$. t(e)$<$now

c. $[\![$ l(was) $]\!]=\lambda P$. P(λe . t(e)$\leq$now)

d. $[\![$ g(l(was)) $]\!]=\lambda Q \lambda x$. Q(x)(λe . t(e)$\leq$now)

e. $[\![$ g(l(was)) g(likely) to win $]\!]=\lambda x \lambda w$. likely(e,w) $\wedge$ pat(e,w') $\wedge$ t(e)$<$now $\wedge$ win(e',w') $\wedge$ ag(e',x) $\wedge$ t(e)$\leq$t(e')

Of course, most raisers don't even allow for-clauses anyways.

(10) a. *It seemed (for) Jill to win.

b. *$[\![$ seem $]\!]=\lambda i \lambda P \lambda w$. seem(e,w) $\wedge$ pat(e,w') $\wedge$ i(e) $\wedge$ P(i')(w')

c. Jill seemed to win.

d. $[\![$ seem $]\!]=\lambda i \lambda P \lambda x \lambda w$. seem(e,w) $\wedge$ pat(e,w') $\wedge$ i(e) $\wedge$ P(x)(i')(w')

2.4 fin-comp $\Rrightarrow$ non-fin-comp

I have generally assumed an ethic of fixing the event as close to the predicate as possible. Then, because the temporal relativizations above make use of ($\lambda e'$. t(e)=t(e')) or ($\lambda e'$. t(e)$\leq$t(e')), defining a totally general function to relate them runs into difficulties. With the functions in sections 2.1, 2.2, and 2.3, it seems to make little difference whether we define them in the "lexicon" or in the "syntax". Perhaps because of my potentially erroneous assumptions, I will have to make a claim like "this function must be defined in the lexicion". That is to say, since I need to introduce another instantiation of a pre-existing constant, it appears almost necessary, and at least most convenient, to define this function as applying before that constant, the event argument, is resolved.[5]

(11) a. $d_R(f)=\lambda e \lambda Q$. f(e)(Q($\lambda e'$. t(e)Rt(e')))

[4] Here, I assume that the adjective merges with its complement before merging with tense.

[5] 'd' stands for "diachronic", since all the other good letters are taken.

b. It is likely that Jill won.

c. $\lambda e \lambda P \lambda i \lambda w$. likely(e,w) ∧ pat(e,w') ∧ i(e) ∧ P(w')

d. It is likely for Jill to win.

e. $\lambda e \lambda Q \lambda i \lambda w$. likely(e,w) ∧ pat(e,w') ∧ i(e) ∧ Q($\lambda e'$. t(e)≤t(e'))(w')

2.5 to

(12) a. ⟦ win ⟧=$\lambda i \lambda x \lambda w$. win(e',w) ∧ i(e') ∧ ag(e',x)

b. ⟦ to win ⟧=$\lambda x \lambda i \lambda w$. win(e',w) ∧ i(e') ∧ ag(e',x)

c. ⟦ to ⟧=$\lambda P \lambda x \lambda i \lambda w$. P(i)(x)(w)

2.6 object before interval

(13) a. ⟦ kiss ⟧=$\lambda i \lambda x \lambda y \lambda w$. kiss(e,w) ∧ ag(e,y) ∧ pat(e,x) ∧ i(e)

b. ⟦ Jill ⟧=$\lambda P \lambda i \lambda y \lambda w$. P(i)(Jill)(y)(w)

c. ⟦ kiss Jill ⟧=$\lambda i \lambda y \lambda w$. kiss(e,w) ∧ ag(e,y) ∧ pat(e,Jill) ∧ i(e)

References

Jacobson, Pauline (1999a) "Binding without pronouns (and pronouns without binding)," To appear in R. Oehrle and G-J. Kruiff (eds.), *Binding and Resource Sensitivity*, Kluwer Academic Press.

Jacobson, Pauline (1999b) "Towards a Variable-Free Semantics," *Linguistics and Philosophy*, 22, 117–184.

Jacobson, Pauline (2000) "Paycheck Pronouns, Bach-Peters Sentences, and Variable-Free Semantics," *Natural Language Semantic*, 8(2), 77–155.

Reinhart, Tanya and Reuland, Eric (1993) "Reflexivity," *Linguistic Inquiry*, 24, 657–720.