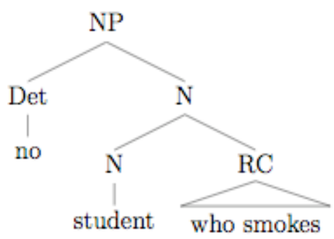
Relative Clauses, Domain Restrictions and Functional NPs

Emma Cunningham, Brown University, SNEWS: 20 Oct 2007

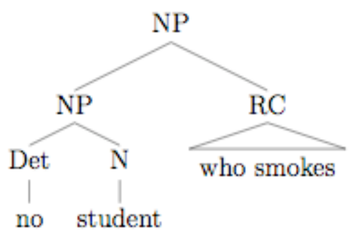
I. Background

No student who smokes is dancing.

Det-Nom



NP-S

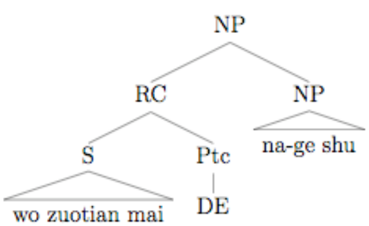


Partee's (1975) argument against NP-S

- Doomed to fail due to the semantics of quantified NPs (Montague 1974)
- E.g. $||\text{no student}|| = \lambda P[\neg\exists x[\text{student}'(x) \,\&\, P(x)]]$
- Det-Nom approach must be the only viable one

Bach & Cooper (1978)

Other languages like Hittite and Chinese (below) require an NP-S analysis

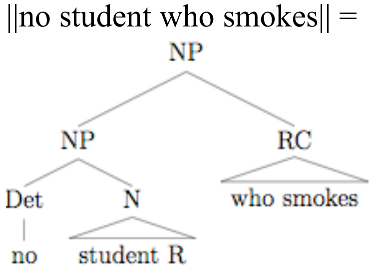


wo zuotian	mai	de	na-ge shu
I yesterday	bought	which	the book

'the book that I bought yesterday'

The Bach & Cooper trick:

$||\text{no student who smokes}|| =$



$||\text{student R}|| = \lambda y[\text{student}'(y) \,\&\, R(y)]$

$||\text{no}||(||\text{student R}||) = \lambda P[\lambda Q[\neg\exists x[P(x) \,\&\, Q(x)]]]\ (\lambda y[\text{student}'(y) \,\&\, R(y)])$

$\qquad = \lambda Q[\neg\exists x[\text{student}'(x) \,\&\, R(x) \,\&\, Q(x)]]$

Bach & Cooper NP-S rule

For an NP-RC structure: $\lambda R[||NP||](||RC||)$

$||\text{no student R who smokes}|| = \lambda R[\lambda Q[\neg\exists x[\text{student}'(x) \,\&\, R(x) \,\&\, Q(x)]]]\ (\lambda y[\text{smokes}'(y)])$

$\qquad = \lambda Q[\neg\exists x[\text{student}'(x) \,\&\, \text{smokes}'(x) \,\&\, Q(x)]]$

R can also reside in the relative clause for stacking

||no student R who smokes R who drinks|| =
||no student R|| = $\lambda Q[\neg\exists x[student'(x) \& R(x) \& Q(x)]]$
||who smokes R|| = $\lambda y[smokes'(y) \& R(y)]$

use NP-S rule for ||no student R who smokes R|| =

$\lambda R[\lambda Q[\neg\exists x[student'(x) \& R(x) \& Q(x)]]] (\lambda y[smokes'(y) \& R(y)])$

$\lambda Q[\neg\exists x[student'(x) \& smokes'(x) \& R(x) \& Q(x)]]$

use NP-S again for ||no student R who smokes R who drinks|| =

$\lambda R[\lambda Q[\neg\exists x[student'(x) \& smokes'(x) \& R(x) \& Q(x)]]] (\lambda y[drinks'(y)])$

$\lambda Q[\neg\exists x[student'(x) \& smokes'(x) \& drinks'(x) \& Q(x)]]$

Bach & Cooper note (fn. 3) that R need not be supplied a relative clause – domain restriction!

II. R

Two ways you could get R in the noun:

1) Just stick it in there! (Stanley & Szabo 2000, Stanley 2002)
   ||student|| = $\lambda R[\lambda x[student'(x) \& R(x)]]$
2) Get it via a shift: $\lambda P[\lambda Q[P \sqcap Q]]$

(why we want the generalized conjunction $\sqcap$ and not ordinary $\cap$ will become clear in the next section)

||student|| = $\lambda x[student'(x)]$
shift: $\lambda Q[\lambda x[student'(x) \& Q(x)]]$

One immediate advantage of the shifting strategy is that we can say it applies to RCs as well.

How to pass up R – geach!

$\mathbf{g}(f_{a,b})$: $\lambda g_{c,a}[\lambda h_c[f((g)(h))]]]$ – curried function composition (Jacobson 1999)

||no|| = $\lambda P[\lambda Q[\neg\exists y[P(y) \& Q(y)]]]$
$\mathbf{g}$(||no||) = $\lambda f_{et,et}[\lambda g_{et}[\lambda Q[\neg\exists y[f((g)(y)) \& Q(x)]]]]$
apply to $\mathbf{r}$(||student||) =

$\lambda f_{et,et}[\lambda g_{et}[\lambda Q[\neg\exists y[f((g)(y)) \& Q(y)]]]] (\lambda R[\lambda x[student'(x) \& R(x)]])$

$\lambda g_{et}[\lambda Q[\neg\exists y[\lambda R[\lambda x[student'(x) \& R(x)]] ((g)(y)) \& Q(y)]]]$

$\lambda g_{et}[\lambda Q[\neg\exists y[student'(y) \& g(y) \& Q(y)]]]$

III. Functional NPs

The woman who every man loves is his mother.

the woman who every man loves is an e,e type function (Jacobson 1994, Sharvit 1999a,b)

Things we will assume (following Jacobson 1994):

- ||the|| is polymorphic and can be of higher order: $\lambda G_{ee,t}[\iota f_{ee}[G(f)]]$
- ||woman|| can also be higher order: $\lambda f_{ee}[\forall x[woman'(f(x))]]$
  - via a shift: $\lambda R[\lambda f[\forall x[R(f(x))]]]$
  - or listed as separate lexical entry

- *The assignment that he gave her that every student most hated every professor for was the last one he gave her*. (Engdahl 1986)
  - Recursive shift rule (Jacobson 2002)

    $\lambda R_{<<e,a>,t>}[\lambda f_{<e,<e,a>>}[\forall x[R(f(x))]]]$
  - ||assignment|| of type e,t: $\lambda x[assign'(x)]$
  - ||assignment|| of type <ee,t>: $\lambda f[\forall x[assign'(f(x))]]$
  - ||assignment|| of type <<e,ee>,t>:

    $\lambda F_{<e,ee>}[\forall y[\lambda f[\forall x[assign'(f(x))]](F(y))]]$

    $\lambda F_{<e,ee>} [\forall y [\forall x[assign'(F(y)(x))]]]$

RCs/domain restrictions modifying functional NPs are functional, too!

Consider the following scenario:

At a music school recital, each student is required to play six pieces.

> A: The piece that every man hates the most is the last one he played.
> B: What about *Hot Cross Buns*?
> A: Of course every man hates that song *the most* but the piece that every man hates the most <u>that he performed</u> is the last one he played.

$||\text{who every man loves}|| = \lambda f_{ee}[\forall z[man'(z) \rightarrow loves'(f(z))(z)]]$

falls out for free given Jacobson's **z** binding operator: $\mathbf{z}(F) = \lambda f[\lambda x[F(f(x))(x)]]$

(see Jacobson 1999)

$\mathbf{z}(loves) = \lambda f[\lambda x[loves'(f(x))(x)]]$

$\mathbf{g}(\text{every man}) = \lambda g[\lambda h[\forall y[man'(y) \rightarrow g(h)(y)]]]$

apply **g**(every man) to **z**(loves) $= \lambda h[\forall y[man'(y) \rightarrow loves'(h(y))(y)]]$

But now things get tricky for our previous attempt to use R to account for relative clauses. Recall our two options:

1) Just stick it in there!

   $||woman|| = \lambda R[\lambda x[woman'(x) \;\&\; R(x)]]$

2) Get it via a shift: $\lambda P[\lambda Q[P \sqcap Q]]$

   $||woman|| = \lambda x[womant(x)]$

   shift: $\lambda Q[\lambda x[woman'(x) \;\&\; Q(x)]]$

The "just stick it in there" approach is stuck with an e,t type restrictor unless you state that "woman" is always ambiguous between an e,t type and a functional interpretation and posit a distinct lexical entry for each interpretation. You would have to since the two readings would be:

$\lambda R_{et}[\lambda x_e[woman'(x) \;\&\; R(x)]]$ $\qquad$ $\lambda G_{eet}[\lambda f_{ee}[\forall z[woman'(f(z))] \;\&\; G(f)]]$

(One possible repair is to geach the shift rule and then apply it to the lower-typed lexical entry:

$\mathbf{g}(\lambda R_{<<e,a>,t>}[\lambda f_{<e,<e,a>>}[\forall x[R(f(x))]]] = \lambda p[\lambda q[\lambda f[\forall x[p(q)(f(x))]]]]$

apply to $||woman\ R|| = \lambda p[\lambda q[\lambda f[\forall x[p(q)(f(x))]]]] (\lambda R_{et}[\lambda y_e[woman'(y) \;\&\; R(y)]])$

$= \lambda q[\lambda f[\forall x[\lambda R_{et}[\lambda y_e[woman'(y) \;\&\; R(y)]] (q)(f(x))]]]$

$= \lambda q[\lambda f[\forall x[woman'(f(x)) \;\&\; q(f(x))]]]$

But then we would be restricting the domain of individuals, not functions!)

The shifting strategy:

$||woman|| = \lambda x_e[woman'(x)]$

shift ||woman|| to functional meaning: $\lambda f_{ee}[\forall z[woman'(f(z))]]$

apply **r** to functional $||woman|| = \lambda P[\lambda Q[P \sqcap Q]] (\lambda f_{ee}[\forall z[woman'(f(z))]])$

$= \lambda Q[\lambda g_{ee}[\forall z[woman'(g(z))] \;\&\; Q(g)]]$

Where the domain restriction is over functions!

The rest of the derivation:
g(the) = $\lambda a[\lambda b[\iota f[a(b)(f)]]]$
apply **g**(the) to **r**(**funct**(woman)) =

$\lambda a[\lambda b[\iota f[a(b)(f)]]](\lambda Q[\lambda g_{ee}[\forall z[\text{woman}'(g(z))] \& Q(g)]])$
$\lambda b[\iota f[\lambda Q[\lambda g_{ee}[\forall z[\text{woman}'(g(z))] \& Q(g)]]\,(b)(f)]]$
$\lambda b[\iota f[\forall z[\text{woman}'(f(z))] \& b(f)]]$

apply to ||who every man loves|| =

$\lambda b[\iota f[\forall z[\text{woman}'(f(z))] \& b(f)]]\,(\lambda h[\forall y[\text{man}'(y) \rightarrow \text{loves}'(h(y))(y)]])$
$\iota f[\forall z[\text{woman}'(f(z))] \& \lambda h[\forall y[\text{man}'(y) \rightarrow \text{loves}'(h(y))(y)]]\,(f)]$
$\iota f[\forall z[\text{woman}'(f(z))] \& \forall y[\text{man}'(y) \rightarrow \text{loves}'(f(y))(y)]]$

the e,e function whose range is the set of women and whose range is the set of loved by every man – e.g. *the-mother-of-him*

IV. Considerations

- not quite an argument <u>for</u> NP-S
- extraposition
- everybody needs a rule for intersection anyways
- relative pronoun as a plain old pronoun – advantage for pied piping (Jacobson 2000)
- exceptive constructions (von Fintel 1993, 2000)

Possible dilemmas:
- lexicalized GQs: *someone*, *nobody*, *everything*, etc.
- syntax??

References:

Bach, E. & Cooper, R. (1978) The NP-S analysis of relative clauses and compositional semantics, *Linguistics and Philosophy* **2**.
Engdahl, E. (1986) *Constituent Questions*, Dordrecht: Reidel.
von Fintel, K. (1993) Exceptive Constructions, *Natural Language Semantics* **1**.
von Fintel, K. (1994) Restrictions on Quantifier Domains, Ph.D. thesis, University of Massachusetts at Amherst.
von Fintel, K. (2000) Exceptive Constructions, Seminar Notes, MIT.
Jacobson, P. (1994) Binding Connectivity in Copular Sentences in M. Harvey and L. Santelmann (eds.), *Proceedings of SALT IV*, Ithaca, NY: Cornell University.
Jacobson, P. (1999) Towards a Variable-Free Semantics, *Linguistics and Philosophy* **22**.
Jacobson, P. (2000) Pied Piping and Reconstruction, Workshop on Relative Clauses, Tel Aviv University.
Jacobson, P. (2002) Direct Compositionality and Variable-Free Semantics: The Case of Binding into Heads, in B. Jackson (ed.), *Proceedings of the 12th Conference on Semantics and Linguistic Theory*, Ithaca, NY: Cornell University CLS Publications.
Montague, R. (1974) The Proper Treatment of Quantification in Ordinary English, in R. Thomason (ed.), *Formal Philosophy: Selected Papers of Richard Montague*, 247-278, New Haven, CT: Yale University Press.
Partee, B. (1975) Montague Grammar and Transformational Grammar, *Linguistic Inquiry* **6**.
Sharvit, Y. (1999a) Connectivity in Specificational Sentences, *Natural Language Semantics* **7**.
Sharvit, Y. (1999b) Functional Relative Clauses, *Linguistics and Philosophy* **22**.
Stanley, J. & Szabo, Z. (2000) On Quantifier Domain Restriction, *Mind and Language* **15**.
Stanley, J. (2002) Nominal Restriction, in Peters & Preyer (eds.), *Logical Form and Language*, Oxford: Oxford University Press.
Steedman, M. (1987) Combinatory Grammars and Parasitic Gaps, *Natural Language and Linguistic Theory* **5**.