John is easy to please: Government-Binding Theory's Empty Categories in a Linear Grammar

Carl Pollard

Department of Linguistics
Ohio State University

ACG@10, LaBRI
Bordeaux
December 9, 2011

Tough-Movement (1/2)

Paradigms like the following have troubled generative grammarians since the mid 1960s:

a. It is easy (for Mary) to please John.

b. John$_i$ is easy (for Mary) to please t$_i$.

- The two sentences mean the same thing: that pleasing John is something that one (or Mary) has an easy time doing.
- It's the (b) version that has been troublesome, because the object of the infinitive, indicated by t, seems to have moved to the subject position of the finite sentence.
- But the syntactic relationship, indicated by coindexation, between the object "trace" and the subject doesn't fall straightforwardly under recognized rule types.

Tough-Movement (2/2)

- As expressed by Hicks (2009), citing Holmberg (2000):

  'Within previous principles-and-parameters models, TCs [tough constructions] have remained "unexplained and in principle unexplainable" because of incompatability with constraints on θ-role assignment, locality, and Case.'
- Hicks, building on a notion of "smuggling" introduced by Collins (2005), proposes a phase-based minimalist analysis in terms of "A-moving a constituent out of a 'complex' null operator that has already undergone $\bar{\text{A}}$-movement."
- This talk sketches a simple analysis of TCs within a a λ-grammar-like framework called **linear grammar** (LG).

This Talk

- We describe LG, a practical, pedagogical, framework for linguistic analysis, and sketch a simple English fragment.
- The fragment covers a range of constructions that were of central interest in the early decades of generative grammar:
  - wh-movement, later subsumed under $\bar{\mathrm{A}}$-movement
  - raising, later subsumed under A-movement
  - control
- GB theory analyzed these constructions in terms of the empty categories (ECs):
  - trace (aka 'syntactic variable').
  - NP-trace
  - PRO
- Once the LG analogs of these ECs have been characterized, the analysis of TCs requires nothing further beyond the lexical entries of the 'tough predicates' themselves.

Wh-Movement/$\bar{\mathrm{A}}$-Movement

Something appears to have moved, possibly long-distance, from a Case-assigned, θ-role-assigned A-position to an $\bar{\mathrm{A}}$ position:

1. Who$_i$ [t$_i$ came]?
2. Who$_i$ did [Mary see t$_i$]?
3. Who$_i$ did [Mary say [John saw t$_i$]]? (long-distance)
4. $*$ Who$_i$ [t$_i$ rained]? (launch site is non-θ)
5. $*$ Who$_i$ did [John try [t$_i$ to come]]? (launch site is non-Case)
6. $*$ Mary told John$_i$ [she liked t$_i$]. (landing site is an A-position)

A = argument (subject or object)

$\bar{\mathrm{A}}$ = nonargument

[. . .] = sentence boundary

Raising

Something seems to have moved from a non-Case, A-position to a superjacent, non-θ, A-position:

1. John$_i$ seems [n$_i$ to be happy].
2. It$_i$ seems [n$_i$ to be raining].
3. $*$ John$_i$ seems [n$_i$ is happy]. (launch site is Case-assigned)
4. $*$ John$_i$ seems [Mary believes [n$_i$ to be happy]]. (landing site is not superjacent)
5. $*$ It$_i$ tries [n$_i$ to be raining]. (landing site is θ-assigned)
6. $*$ Who$_i$ does [John seem [n$_i$ to be happy]]? (landing site is an $\bar{\text{A}}$-position)

Control

An EC in a θ-assigned non-Case position seems to be anaphoric to something in a superjacent A-position:

1. Mary$_i$ tries [PRO$_i$ to be happy].
2. $*$ Mary$_i$/it$_i$ tries [PRO$_i$ to rain]. (EC is in a non-θ position.)
3. $*$ John tries [Mary to like PRO$_i$]. (EC is in a Case position)
4. $*$ Mary$_i$ tries [John believes [PRO$_i$ to be happy]]. (landing site is not superjacent)
5. $*$ Who$_i$ did [John try [PRO$_i$ to be happy]]? (landing site is an $\bar{\text{A}}$-position)

What's Tough about *Tough*-'Movement'

- Like $\bar{A}$-movement, the launch site is a θ-assigned Case position, and it can be long-distance:

  a. John$_i$ is easy for Mary [to please t$_i$].
  b. John$_i$ is easy for Mary [to get other people [to distrust t$_i$]].

- Like A-movement, the landing site is a non-θ A-position.
- Like Control, the 'antecedent' of the EC must be 'referential', i.e. it can't be a dummy or an idiom chunk:

  a. John is easy to believe to be bluffing.
  b. $*$ It is easy to believe to be raining.
  c. $*$ There is easy to believe to be a largest prime number.
  d. $*$ The shit is easy to believe to have hit the fan. (no idiomatic interpretation)

Linear Grammar Preview

LG is similar to λ-grammar, with these differences:

- As in pregroup grammar, the basic tectotypes are ordered.
- Hypothesis axiom schema is correspondingly generalized.
- No function from tectotypes to semantic types.
- In the simplest case (as here), the pheno theory is just the HO theory of monoids, with one basic type s (string).
- The semantic theory can be:
  - static (as here) or dynamic
  - hyperintensional (as here), intensional, or extensional

Some Analytic Assumptions of this Fragment

- 'Traces' (hypotheses) are restricted to have pheno-type s.
- Unlike finite VPs, nonfinite VPs and predicatives have phenotype s, not s $\rightarrow$ s.
- No morphosyntactic features (e.g. via dependent typing), instead just lots of basic tectotypes.
- Lexical entries written so that phenogrammatically leftmost arguments are consumed first (not important here, but shortens derivations with dynamic semantics).
- Only third-singular NPs included, for expository simplicity, but adding person/number agreement is unproblematic.

Linear Grammar Overview

- An LG for an NL is a sequent-style ND system that recursively defines a set of ordered triples called **signs**, each of which is taken to represent an expression of the NL.
- Signs are notated:

$$a : A; B; c : C$$

  where

  - $a : A$, the **pheno**, is a typed term of the pheno theory
  - B, the **tecto**, is a formula of the linear tecto logic
  - $c : C$, the **semantics**, is a typed term of the semantic theory

The Pheno Theory

- logical basic types: T (unit) and t (truth values)
- nonlogical basic type: s (string)
- Nonlogical constants:
  - $\mathbf{e} : \mathrm{s}$ (null string)
  - string constants for phenos of lexical signs, e.g. it, is, easy, for, mary, to, please, john
  - $\cdot : \mathrm{s} \to \mathrm{s} \to \mathrm{s}$ (concatenation, written infix)
- Nonlogical axioms (here $s, t, u : \mathrm{s}$):

$$\vdash \forall_{stu}.(s \cdot t) \cdot u = s \cdot (t \cdot u)$$
$$\vdash \forall_{s}.(\mathbf{e} \cdot s) = s$$
$$\vdash \forall_{s}.(s \cdot \mathbf{e}) = s$$

i.e. the strings form a monoid with concatenation as the associative operation and null string as identity.

Some Basic Tectos

Nom (nominative, e.g. *he*, *she*)

Acc (accusative, e.g. *him*, *her*)

For (*for*-phrase, e.g. *for Mary*)

It ('dummy pronoun' *it*)

S (finite clause)

Inf (infinitive clause)

Bse (base clause)

Prd (predicative clause)

PrdA (adjectival predicative clause)

More Basic Tectos

- Neu (case-neutral, e.g. *John*, *Mary*)
- PRO (LG counterpart of GB's PRO)

  Used for subject of nonfinite verbs and predicatives that assign a semantic role to the subject, e.g. nonfinite *please*
- NP (LG counterpart of GB's NP-trace)

  Used for subject of nonfinite verbs and predicatives that don't assign a semantic role to the subject, e.g. nonfinite *seem*, infinitive *to*
- NOM (generalized nominatives)

  Used for subject of finite verbs that don't assign a semantic role to the subject, e.g. *seems*, *is*
- ACC (generalized accusatives)

  Used for objects of verbs that don't assign a semantic role to the object, e.g. infinite-complement-*believe*

Ordering of Basic Tectos

Neu $<$ Nom

Neu $<$ Acc

Nom $<$ PRO

Acc $<$ PRO

Nom $<$ NOM

Acc $<$ ACC

It $<$ NOM

It $<$ ACC

NOM $<$ NP

ACC $<$ NP

PRO $<$ NP

PrdA $<$ Prd

The Semantic Theory

- Logical basic types: T (unit) and t (truth values)
- Basic types: e (individuals) and p (propositions).
- For convenience, we abbreviate certain types as follows:

  a. $\mathrm{p}_{\mathit{0}} =_{\mathrm{def}} \mathrm{p}$
  b. $\mathrm{p}_{n+\mathit{1}} =_{\mathrm{def}} \mathrm{e} \to \mathrm{p}_n$

- Logical constant: $* : \mathrm{T}$, used for vacuous meanings (e.g. of dummy pronous)
- Nonlogical constants used for lexical meanings (next slide).
- Nonlogical rules (not needed here) are analogous to meaning postulates in Montague semantics.

Some Nonlogical Constants for Lexical Semantics

$\vdash \mathsf{j} : \mathrm{e}$ (John)

$\vdash \mathsf{m} : \mathrm{e}$ (Mary)

$\vdash \mathsf{rain} : \mathrm{p}$

$\vdash \mathsf{please} : \mathrm{p}_2$

$\vdash \mathsf{easy} : \mathrm{e} \rightarrow \mathrm{p}_1 \rightarrow \mathrm{p}$

LG Architecture

In its simplest form (as here), an LG consists of:

- Two kinds of **axioms**:
  - **logical** axioms, called **traces**
  - **nonlogical** axioms, called **lexical entries**
- Two rule schemas:
  - Modus Ponens (MP)
  - Hypothetical Proof (HP)

Before considering the precise form of the axioms and rules, we need to discuss the form of LG **sequents**.

LG Sequents

- A sign is called **hypothetical** provided its pheno and semantics are both variables.
- An LG **sequent** is an ordered pair whose first component (the **context**) is a finite multiset of hypothetical signs, and whose second component (the **statement**) is a sign.
- The hypothetical sign occurrences in the context are called the **hypotheses** or **assumptions** of the sequent.
- We require that no two hypotheses have the same pheno variable, and that no two hypotheses have the same semantic variable.
- So actually the multisets are sets.

Notational convention: we omit the types of tecto and semantic terms when no confusion will result.

The Trace Axiom Schema

The usual Hypothesis axiom scheme is generalized, corresponding to the ordering of the basic tectos:

Full form:

$$s : \mathrm{s}; B; z : C \vdash s : \mathrm{s}; B'; z : C \ (B \leq B')$$

Short form (when types of variables are known):

$$s; B; z \vdash s; B'; z \ (B \leq B')$$

Uses of these axioms are the LG counterpart of GB traces.

Two Lexical Entries

$\vdash \text{it}; \text{It}; *$ (dummy pronoun *it*)

$\vdash \lambda_s.s \cdot \text{rains}; \text{It} \multimap \text{S}; \lambda_o.\mathsf{rain}$ (o is of type T)

The Two LG Rule Schemas (Full Form)

- Modus Ponens (LG counterpart of Merge)

$$\frac{\Gamma \vdash f : A \to D; B \multimap E; g : C \to F \qquad \Delta \vdash a : A; B; c : C}{\Gamma, \Delta \vdash f\ a : D; E; g\ c : F}\,\mathrm{MP}$$

- Hypothetical Proof (LG counterpart of Move)

$$\frac{\Gamma, x : A; B; z : C \vdash d : D; E; f : F}{\Gamma \vdash \lambda_x.d : A \to D; B \multimap E; \lambda_z.f : C \to F}\,\mathrm{HP}$$

The Two LG Rule Schemata (Short Form)

These forms are used when the types of the terms are known.

- Modus Ponens

$$\frac{\Gamma \vdash f; B \multimap E; g \qquad \Delta \vdash a; B; c}{\Gamma, \Delta \vdash f\ a; E; g\ c}\ \text{MP}$$

- Hypothetical Proof

$$\frac{\Gamma, x; B; z \vdash d; E; f}{\Gamma \vdash \lambda_x.d; B \multimap E; \lambda_z.f}\ \text{HP}$$

N.B.: By convention, the label MP is omitted.

Three Useful Derived LG Rule Schemas

These rules are schematized over B, B' with $B \leq B'$.

- Derived Rule Schema 1

$$\frac{\Gamma \vdash a; B; c}{\Gamma \vdash a; B'; c}\,\text{D1}$$

- Derived Rule Schema 2

$$\frac{\Gamma \vdash f; B' \multimap A; g}{\Gamma \vdash f; B \multimap A; g}\,\text{D2}$$

- Derived Rule Schema 3

$$\frac{\Gamma \vdash f; A \multimap B; g}{\Gamma \vdash f; A \multimap B'; g}\,\text{D3}$$

An LG Proof

Unsimplified:

$$\frac{\vdash \lambda_s.s \cdot \mathrm{rains}; \mathrm{It} \multimap \mathrm{S}; \lambda_o.\mathsf{rain} \qquad \vdash \mathrm{it}; \mathrm{It}; *}{\vdash (\lambda_s.s \cdot \mathrm{rains})\ \mathrm{it}; \mathrm{S}; (\lambda_o.\mathsf{rain})\ *}$$

Simplified:

$$\frac{\vdash \lambda_s.s \cdot \mathrm{rains}; \mathrm{It} \multimap \mathrm{S}; \lambda_o.\mathsf{rain} \qquad \vdash \mathrm{it}; \mathrm{It}; *}{\vdash \mathrm{it} \cdot \mathrm{rains}; \mathrm{S}; \mathsf{rain}}$$

We use provable equalities of the semantic theory to simplify terms in intermediate conclusions before using them as premisses for subsequent rule instances.

More Lexical Entries

$\vdash \text{john}; \text{Neu}; \mathsf{j}$

$\vdash \text{mary}; \text{Neu}; \mathsf{m}$

$\vdash \lambda_{st}.s \cdot \text{pleases} \cdot t; \text{Nom} \multimap \text{Acc} \multimap \text{S}; \mathsf{please}$

$\vdash \lambda_{t}.\text{please} \cdot t; \text{Acc} \multimap \text{PRO} \multimap \text{Bse}; \lambda_{yx}.\mathsf{please}\ x\ y$

$\vdash \lambda_{t}.\text{to} \cdot t; (A \multimap \text{Bse}) \multimap A \multimap \text{Inf}; \lambda_{P}.P\ (A \leq \text{NP},\ P : B \rightarrow \text{p})$

$\vdash \lambda_{st}.s \cdot \text{is} \cdot t; A \multimap (A \multimap \text{Prd}) \multimap \text{S}; \lambda_{xP}.P\ x\ (A \leq \text{NOM}, x : B, P : B \rightarrow \text{p})$

$\vdash \lambda_{t}.\text{for} \cdot t; \text{Acc} \multimap \text{For}; \lambda_{x}.x$

$\vdash \lambda_{st}.\text{easy} \cdot s \cdot t; \text{For} \multimap (\text{PRO} \multimap \text{Inf}) \multimap \text{It} \multimap \text{PrdA}; \lambda_{xPo}.\mathsf{easy}\ x\ P$

$\vdash \lambda_{sf}.\text{easy} \cdot s \cdot (f\ \mathbf{e}); \text{For} \multimap (\text{Acc} \multimap \text{PRO} \multimap \text{Inf}) \multimap \text{PRO} \multimap \text{PrdA};$
$\lambda_{xry}.\mathsf{easy}\ x\ (r\ y)$

How Neutral Expressions Get Case

$$\dfrac{\dfrac{\vdash \lambda_{st}.s \cdot \text{pleases} \cdot t; \text{Nom} \multimap \text{Acc} \multimap \text{S} \qquad \dfrac{\vdash \text{john}; \text{Neu}; \mathsf{j}}{\vdash \text{john}; \text{Nom}; \mathsf{j}}\,\text{D1}}{\vdash \lambda t.\text{john} \cdot \text{pleases} \cdot t; \text{Acc} \multimap \text{S}; \mathsf{please\ j}} \qquad \dfrac{\vdash \text{mary}; \text{Neu}; \mathsf{m}}{\vdash \text{mary}; \text{Acc}; \mathsf{m}}\,\text{D1}}{\vdash \text{john} \cdot \text{pleases} \cdot \text{mary}; \text{S}; \mathsf{please\ j\ m}}$$

Nonpredicative “Prepositional” Phrases

Here and henceforth, leaves with overbars were already proved as lemmas in earlier derivations.

$$\dfrac{\vdash \lambda_t.\text{for} \cdot t; \text{Acc} \multimap \text{For}; \lambda_x.x \qquad \overline{\vdash \text{mary}; \text{Acc}; \mathsf{m}}}{\vdash \text{for} \cdot \text{mary}; \text{For}; \mathsf{m}}$$

There’s no empirical justification for calling nonpredicative For-phrases ‘prepositional’, so we just treat For as a basic tecto.

An Infinitive Phrase

$$
\dfrac{\vdash \lambda_t.\text{to} \cdot t; (A \multimap \text{Bse}) \multimap A \multimap \text{Inf}; \lambda_P.P \qquad \dfrac{\vdash \lambda_t.\text{please} \cdot t; \text{Acc} \multimap \text{PRO} \multimap \text{Bse} \qquad \dfrac{}{\vdash \text{john}; \text{Acc}; \mathsf{j}}}{\vdash \text{please} \cdot \text{john}; \text{PRO} \multimap \text{Bse}; \mathsf{please\ j}}}{\vdash \text{to} \cdot \text{please} \cdot \text{john}; \text{PRO} \multimap \text{Inf}; \mathsf{please\ j}}
$$

- Here A (two occurrences) in the *to* schema was instantiated as PRO (and B in $P : B \to \text{p}$ as e). This is legitimate because the schematization is over $A \leq \text{NP}$, and in fact $\text{PRO} < \text{NP}$.
- This is an instance of (the LG counterpart of) Raising, in this case of PRO from the base-form complement *please John* to the infinite phrase.
- There is no *sign* of tectotype PRO that 'raises'!

An Impersonal Predicative Phrase

$$\dfrac{\vdash \lambda_{st}.\text{easy} \cdot s \cdot t; \text{For} \multimap (\text{PRO} \multimap \text{Inf}) \multimap \text{It} \multimap \text{PrdA}; \lambda_{xPo}.\mathsf{easy}\ x\ P \qquad \dfrac{}{\vdash \text{for} \cdot \text{mary}; \text{For}; \mathsf{m}}}{\vdash \lambda_t.\text{easy} \cdot \text{for} \cdot \text{mary} \cdot t; (\text{PRO} \multimap \text{Inf}) \multimap \text{It} \multimap \text{PrdA}; \lambda_{Po}.\mathsf{easy}\ m}$$

$$\dfrac{\vdash \lambda_t.\text{to} \cdot t; (A \multimap \text{Bse}) \multimap A \multimap \text{Inf}; \lambda_P.P \qquad \dfrac{\vdash \lambda_t.\text{please} \cdot t; \text{Acc} \multimap \text{PRO} \multimap \text{Bse} \qquad \dfrac{}{\vdash \text{john}; \text{Acc}; \mathsf{j}}}{\vdash \text{please} \cdot \text{john}; \text{PRO} \multimap \text{Bse}; \lambda_x \mathsf{please}\ x\ \mathsf{j}}}{\vdash \text{to} \cdot \text{please} \cdot \text{john}; \text{PRO} \multimap \text{Inf}; \lambda_x.\mathsf{please}\ x\ \mathsf{j}}$$

$$\dfrac{\dfrac{\dfrac{}{\vdash \text{easy} \cdot \text{for} \cdot \text{mary}; (\text{PRO} \multimap \text{Inf}) \multimap \text{It} \multimap \text{PrdA}} \qquad \dfrac{}{\vdash \text{to} \cdot \text{please} \cdot \text{john}; \text{PRO} \multimap \text{Inf}}}{\vdash \text{easy} \cdot \text{for} \cdot \text{mary} \cdot \text{to} \cdot \text{please} \cdot \text{john}; \text{It} \multimap \text{PrdA}; \lambda_o.\mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ \mathsf{j})}}{\vdash \text{easy} \cdot \text{for} \cdot \text{mary} \cdot \text{to} \cdot \text{please} \cdot \text{john}; \text{It} \multimap \text{Prd}; \lambda_o.\mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ \mathsf{j})}\ \text{D3}$$

- This is just like a Control construction, e.g. *Mary tries to please John*, which means $\mathsf{try}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ \mathsf{j})$ ...
- Except that the controller is the For-phrase, rather than the subject (which is only a dummy)
- This semantics of Control (where the infinitive complement is analyzed as a property rather than a proposition) originates with Chierchia (1980's).

It is easy for Mary to please John

$$\frac{\vdash \lambda_{st}.s \cdot \mathrm{is} \cdot t; A \multimap (A \multimap \mathrm{Prd}) \multimap \mathrm{S}; \lambda_{xP}.P\ x \qquad \vdash \mathrm{it}; \mathrm{It}; *}{\vdash \lambda_t.\mathrm{it} \cdot \mathrm{is} \cdot t; (\mathrm{It} \multimap \mathrm{Prd}) \multimap \mathrm{S}; \lambda_P.P\ *}$$

$$\frac{\overline{\vdash \lambda_t.\mathrm{it} \cdot \mathrm{is} \cdot t; (\mathrm{It} \multimap \mathrm{Prd}) \multimap \mathrm{S}} \qquad \overline{\vdash \mathrm{easy} \cdot \mathrm{for} \cdot \mathrm{m} \cdot \mathrm{to} \cdot \mathrm{please} \cdot \mathrm{j}; \mathrm{It} \multimap \mathrm{Prd}; \lambda_o.\mathsf{easy\ m}\ (\lambda_x.\mathsf{please}\ x\ \mathsf{j})}}{\vdash \mathrm{it} \cdot \mathrm{is} \cdot \mathrm{easy} \cdot \mathrm{for} \cdot \mathrm{mary} \cdot \mathrm{to} \cdot \mathrm{please} \cdot \mathrm{john}; \mathrm{S}; \mathsf{easy\ m}\ (\lambda_x.\mathsf{please}\ x\ \mathsf{j})}$$

- Here A in *is* is instantiated as It (and B in $x : B$ as T, so $P : \mathrm{T} \to \mathrm{p}$).
- This is another instance of 'Raising', in this case of the unrealized It subject of the predicative phrase *easy for Mary to please John* to the sentence.
- In no sense was the sign *it* ever in the predicative phrase.

A Gappy Infinitive Phrase

$$\frac{\vdash \lambda_t.\text{please} \cdot t; \text{Acc} \multimap \text{PRO} \multimap \text{Bse}; \lambda_{yx}.\mathsf{please}\ x\ y \qquad s; \text{Acc}; y \vdash s; \text{Acc}; y}{s; \text{Acc}; y \vdash \text{please} \cdot s; \text{PRO} \multimap \text{Bse}; \lambda_x.\mathsf{please}\ x\ y}$$

$$\cfrac{\cfrac{\vdash \lambda_t.\text{to} \cdot t; (A \multimap \text{Bse}) \multimap A \multimap \text{Inf}; \lambda_P.P \qquad \cfrac{}{s; \text{Acc}; y \vdash \text{please} \cdot s; \text{PRO} \multimap \text{Bse}; \lambda_x.\mathsf{please}\ x\ y}}{s; \text{Acc}; y \vdash \text{to} \cdot \text{please} \cdot s; \text{PRO} \multimap \text{Inf}; \lambda_x.\mathsf{please}\ x\ y}}{\vdash \lambda_s.\text{to} \cdot \text{please} \cdot s; \text{Acc} \multimap \text{PRO} \multimap \text{Inf}; \lambda_{yx}.\mathsf{please}\ x\ y}\ \text{HP}$$

- The object trace, which is withdrawn in the last proof step, captures the sense in which '*Tough*-Movement' works like an $\bar{\text{A}}$ (long-distance) dependency.
- The λ_s and λ_y in the pheno and semantics of the conclusion are prefigured by the empty operator binding the trace in Chomsky's (1977) analysis of this same construction:

$$[\mathsf{john}_i\ \mathsf{is\ easy}\ \text{O}_i[\text{PRO}\ \mathsf{to\ please}\ \text{t}_i]]$$

- Unlike Hicks' analysis, there is nothing 'complex' about the operator that binds the trace (it is just λ), and no sense in which anything ever 'moves out' of it.

A Personal Predicative Phrase

$$\dfrac{\vdash \lambda_{sf}.\text{easy} \cdot s \cdot (f\ \mathbf{e}); \text{For} \multimap (\text{Acc} \multimap \text{InfP}) \multimap \text{PRO} \multimap \text{PrdA}; \lambda_{xRy}.\mathsf{easy}\ x\ (R\ y) \qquad \overline{\vdash \text{for} \cdot \text{mary}; \text{For}; \mathsf{m}}}{\vdash \lambda_f.\text{easy} \cdot \text{for} \cdot \text{mary} \cdot (f\ \mathbf{e}); (\text{Acc} \multimap \text{InfP}) \multimap \text{PRO} \multimap \text{PrdA}; \lambda_{Ry}.\mathsf{easy}\ m\ (R\ y)}$$

$$\dfrac{\overline{\vdash \lambda_f.\text{easy} \cdot \text{for} \cdot \text{mary} \cdot (f\ \mathbf{e}); (\text{Acc} \multimap \text{InfP}) \multimap \text{PRO} \multimap \text{PrdA}} \qquad \overline{\vdash \lambda_s.\text{to} \cdot \text{please} \cdot s; \text{Acc} \multimap \text{InfP}}}{\dfrac{\dfrac{\vdash \text{easy} \cdot \text{for} \cdot \text{mary} \cdot \text{to} \cdot \text{please}; \text{PRO} \multimap \text{PrdA}; \lambda_y.\mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ y)}{\vdash \text{easy} \cdot \text{for} \cdot \text{mary} \cdot \text{to} \cdot \text{please}; \text{Nom} \multimap \text{PrdA}; \lambda_y.\mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ y)}\ \text{D2}}{\vdash \text{easy} \cdot \text{for} \cdot \text{mary} \cdot \text{to} \cdot \text{please}; \text{Nom} \multimap \text{Prd}; \lambda_y.\mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ y)}\ \text{D3}}$$

- Here ‘InfP’ abbreviates PRO $\multimap$ Inf.

John is easy for Mary to please

$$\dfrac{\vdash \lambda_{st}.s \cdot \text{is} \cdot t; A \multimap (A \multimap \text{Prd}) \multimap \text{S}; \lambda_{xP}.P\ x \qquad \dfrac{}{\vdash \text{john}; \text{Nom}; \mathsf{j}}}{\vdash \lambda_t.\text{john} \cdot \text{is} \cdot t; (\text{Nom} \multimap \text{Prd}) \multimap \text{S}; \lambda_P.P\ \mathsf{j}}$$

$$\dfrac{\dfrac{}{\vdash \lambda_t.\text{john} \cdot \text{is} \cdot t; (\text{Nom} \multimap \text{Prd}) \multimap \text{S}; \lambda_P.P\ \mathsf{j}} \qquad \dfrac{}{\vdash \text{easy} \cdot \text{for} \cdot \text{mary} \cdot \text{to} \cdot \text{please}; \text{Nom} \multimap \text{Prd}; \lambda_y.\mathsf{easy}\ \mathsf{m}}}{\vdash \text{John} \cdot \text{is} \cdot \text{easy} \cdot \text{for} \cdot \text{mary} \cdot \text{to} \cdot \text{please}; \text{S}; \mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ \mathsf{j})}$$

- Here A in *is* is instantiated as Nom.
- This is another instance of 'Raising', in this case of the unrealized Nom subject of the predicative phrase *easy for Mary to please* to the sentence.
- But *John* was never actually in the predicative phrase!

Conclusion

- The properties of the so-called *Tough*-construction don't necessitate any revision to linguistic theory (as long as you have the right theory to start with).
- They are simply consequences of the lexical entries for the so-called *Tough*-predicates, such as

  $\vdash \lambda_{sf}.\mathrm{easy} \cdot s \cdot (f\ \mathbf{e}); \mathrm{For} \multimap (\mathrm{Acc} \multimap \mathrm{PRO} \multimap \mathrm{Inf}) \multimap \mathrm{PRO} \multimap \mathrm{PrdA}; \lambda_{xry}.\mathsf{easy}\ x\ (r\ y)$

- The entire derivation of sentences with such predicates is just business as usual.
- What's so tough about that?

Ongoing Research in or on Linear Grammar

- Crosslinguistic study of comparative correlative constructions: Elizabeth Smith
- Wh-'movement' and cliticization in Bosnian-Croation-Serbian: Vedrana Mihalicek
- Syntax, semantics, and prosody of focus in K'iche': Murat Yasavul
- Incorporating intonation and information-structural meaning into LG: Chris Worth
- Weak familiarity and conventional implicature in DyCG (= LG + hyperintensional dynamic semantics): Scott Martin
- Projective entailments in DyCG: Carl Pollard and Elizabeth Smith
- Evidentiality in Tagalog: Greg Kierstead