Linear Grammar Basics

Carl Pollard

Department of Linguistics
Ohio State University

June 13, 2012

Linear Grammar Overview

- A linear grammar (LG) for a language 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 language.
- Signs are notated in the form

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

  where

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

LG Architecture

In its simplest form, an LG consists of:

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

  A (very) few more rules will be added in due course.

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 or the same semantic variable.
- So the contexts are actually just finite **sets**.

Note: we omit pheno and semantic types when they can be inferred from the corresponding terms.

The Trace Axiom Schema

Full form:

$$s : \mathrm{s}; B; z : C \vdash s : \mathrm{s}; B; z : C$$

Short form (when types of variables are known):

$$s; B; z \vdash s; B; z$$

Note: The pheno is required to be of type s. This is not logically necessary, but is empirically motivated.

Two Lexical Entries to Get Started

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

$\vdash \lambda_s.s \cdot \text{rained}; \text{It} \multimap \text{S}; \lambda_o.\mathsf{rain} : \text{T} \rightarrow \text{p}$

The Two LG Rule Schemas (Full Form)

- Modus Ponens

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

- Hypothetical Proof

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

The Two LG Rule Schemata (Short Form)

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

- Modus Ponens

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

- Hypothetical Proof

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

An LG Proof

Unsimplified:

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

Simplified:

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

We use provable equalities (most often, rule (β)) of the pheno and semantic theories to simplify terms in intermediate conclusions before using them as premisses for later rule instances.

More Lexical Entries

$\vdash$ pedro; NP; p

$\vdash$ chiqita; NP; c

$\vdash$ maria; NP; m

$\vdash \lambda_s.s \cdot \text{brayed}; \text{NP} \multimap \text{S}; \mathsf{bray}$

$\vdash \lambda_{st}.s \cdot \text{beat} \cdot t; \text{NP} \multimap \text{NP} \multimap \text{S}; \mathsf{beat}$

$\vdash \lambda_{stu}.s \cdot \text{gave} \cdot t \cdot u; \text{NP} \multimap \text{NP} \multimap \text{NP} \multimap \text{S}; \mathsf{give}$

$\vdash \lambda_{st}.s \cdot \text{believed} \cdot t; \text{NP} \multimap \bar{\text{S}} \multimap \text{S}; \mathsf{believe}$

Note: The finite verb entries are written to combine the verb first with the subject, then with the complements (the reverse of how things are traditionally done!)

Still More Lexical Entries

$\vdash \text{donkey}; \mathrm{N}; \mathsf{donkey}$

$\vdash \text{farmer}; \mathrm{N}; \mathsf{farmer}$

$\vdash \lambda_s.\text{that} \cdot s; \mathrm{S} \multimap \bar{\mathrm{S}}; \lambda_p.p$ (complementizer *that*)

$\vdash \lambda_{fs}.s \cdot \text{that} \cdot (f\ \mathbf{e}); (\mathrm{NP} \multimap \mathrm{S}) \multimap \mathrm{N} \multimap \mathrm{N}; \mathsf{that}$ (relativizer *that*)

$\vdash \lambda_{sf}.f\ (\text{every} \cdot s); \mathrm{N} \multimap \mathrm{QP}; \mathsf{every}$

$\vdash \lambda_{sf}.f\ (\text{some} \cdot s); \mathrm{N} \multimap \mathrm{QP}; \mathsf{some}$

- 'QP' (for 'quantified NP') abbreviates $(\mathrm{NP} \multimap \mathrm{S}) \multimap \mathrm{S}$.
- The motivation for giving these expressions a 'raised' tecto type is semantic and will be explained in due course.

Another LG Proof

$$\frac{\vdash \lambda_s.s \cdot \text{brayed}; \text{NP} \multimap \text{S}; \mathsf{bray} \qquad \vdash \text{chiqita}; \text{NP}; \mathsf{c}}{\vdash \text{chiqita} \cdot \text{brayed}; \text{S}; \mathsf{bray\ c}}$$

Yet Another LG Proof

$$\cfrac{\cfrac{\vdash \lambda_{st}.s \cdot \text{beat} \cdot t; \text{NP} \multimap \text{NP} \multimap \text{S}; \textsf{beat} \qquad \vdash \text{pedro}; \text{NP}; \textsf{p}}{\lambda_t.\text{pedro} \cdot \text{beat} \cdot t; \text{NP} \multimap \text{S}; \textsf{beat p}} \qquad \vdash \text{chiqita}; \text{NP}; \textsf{c}}{\text{pedro} \cdot \text{beat} \cdot \text{chiquita}; \text{S}; \textsf{beat p c}}$$

Note that we had to shrink this to tiny to fit it on the slide!

The Same Proof with Semantics Omitted

Alternatively, if we are not concerned about semantics, we can sometimes overcome the space problem by omitting the semantics components of the signs:

$$\dfrac{\dfrac{\vdash \lambda_{st}.s \cdot \mathrm{beat} \cdot t; \mathrm{NP} \multimap \mathrm{NP} \multimap \mathrm{S} \qquad \vdash \mathrm{pedro}; \mathrm{NP}}{\lambda_t.\mathrm{pedro} \cdot \mathrm{beat} \cdot t; \mathrm{NP} \multimap \mathrm{S}} \qquad \vdash \mathrm{chiqita}; \mathrm{NP}}{\mathrm{pedro} \cdot \mathrm{beat} \cdot \mathrm{chiquita}; \mathrm{S}}$$

This approach has its limits.

An Oversized LG Proof

$\vdash \lambda_s.s \cdot \text{brayed}; \text{N}$

$\vdash \lambda_{st}.s \cdot \text{believed} \cdot t; \text{NP} \multimap \bar{\text{S}} \multimap \text{S} \qquad \vdash \text{pedro}; \text{NP} \qquad \vdash \lambda_s.\text{that} \cdot s; \text{S} \multimap \bar{\text{S}} \qquad \vdash \text{chic}$

$\vdash \lambda_t.\text{pedro} \cdot \text{believed} \cdot t; \bar{\text{S}} \multimap \text{S} \qquad \vdash \text{that} \cdot \text{chiquita} \cdot \text{brayed}; \bar{\text{S}}$

$\vdash \text{pedro} \cdot \text{believed} \cdot \text{that} \cdot \text{chiquita} \cdot \text{brayed}; \text{S}$

Another Solution to the Space Problem

[1]:

$$\frac{\vdash \lambda_{st}.s \cdot \text{believed} \cdot t; \text{NP} \multimap \bar{\text{S}} \multimap \text{S}; \textsf{believe} \qquad \vdash \text{pedro}; \text{NP}; \textsf{p}}{\vdash \lambda_t.\text{pedro} \cdot \text{believed} \cdot t; \bar{\text{S}} \multimap \text{S}; \textsf{believe p}}$$

[2]:

$$\frac{\vdash \lambda_s.\text{that} \cdot s; \text{S} \multimap \bar{\text{S}}; \lambda_p.p \qquad \dfrac{\vdash \lambda_s.s \cdot \text{brayed}; \text{NP} \multimap \text{S}; \textsf{bray} \qquad \vdash \text{chiqita}; \text{NP}; \textsf{c}}{\vdash \text{chiquita} \cdot \text{brayed}; \text{S}; \textsf{bray c}}}{\vdash \text{that} \cdot \text{chiquita} \cdot \text{brayed}; \bar{\text{S}}; \textsf{bray c}}$$

$$\frac{[1] \qquad [2]}{\vdash \text{pedro} \cdot \text{believed} \cdot \text{that} \cdot \text{chiquita} \cdot \text{brayed}; \text{S}; \textsf{believe p (bray c)}}$$

Starting to Get Real: English ‘NPs’

- To get started, we assumed tectos NP (for names) and It (for dummy *it*), but this is too simple.
- Even if we consider only third person singular noun phrases, we still must account for these facts:
  - Besides dummy *it*, there is also dummy *there*, which has a completely different distribution
  - Names and NPs formed by combining a determiner with a common noun occur both as subject and as object of verb or preposition.
  - The same is true of the dummy pronouns.
  - But, except for nonhuman *it*, definite pronouns have different forms, of which some (*he, she*) can’t be objects and others (*her, him*) can’t be subjects.
  - Only a few verbs, e.g. *be* and *seem*, allow dummy subjects; and only a few, e.g. *believe*, allow dummy objects.

Are Features Necessary?

- In most syntactic frameworks (CCG, HPSG, LFG, MP) problems of this kind are addressed through the use of **features**, also called **attributes**.
- For example, in HPSG, NPs specify values for the features CASE and NFORM.
- But in a framework (such as LG) based on proof theory, it's unclear what 'features' would be: logical formulas aren't usually thought of as having 'features'.
- We'll use a different approach due to Lambek (1999) in the context of his framework called **pregroup grammar**.
- Pregroup grammar is based on classical **bilinear** logic, but Lambek's idea works just as well with linear logic.

Preordering the Basic Tectos (1/2)

- Lambek proposed *preordering* the basic syntactic types.
- The basic intuition is that if $A \leq B$, then any sign with tecto A can also be considered as a sign with tecto B.
- In this case we say A is a **(tecto) subtype** of B.
- For example: we would like to say that the tecto of 'NPs' which can serve as both subjects and objects (which we will call Neu, for 'neutral') is a subtype of the tecto of 'NPs' that can serve as subjects (which we will call Nom, for 'nominative').

Preordering the Basic Tectos (2/2)

- In the grammar, we directly assert certain inequalities, such as Neu $\leq$ Nom, and then define $\leq$ to be the smallest preorder (reflexive transitive relation) on basic tectos that includes all the asserted inequalities.
- Then we revise the Trace axiom schema to the following more general form (the original schema corresponds to the instances where $B = B'$):

  Trace Axiom Schema (Generalized):

$$x; B; z \vdash x; B'; z \text{ (for } B \leq B'\text{)}$$

Three Derived Rule Schemas

These schemas (schematized over $B \leq B'$) are very useful for shortening LG proofs. (Their derivations are left as exercises.)

- 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

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

Preordering Basic Tectos to Analyze English Case

- For now we only consider sentences with finite verbs.
- Later we'll elaborate our approach to handle issues about 'unrealized' subjects of nonfinite verb forms (base forms, infinitives, and participles) and of nonverbal 'predicative' expressions (predicative NPs, APs, and PPs).
- First we discard the tecto NP and replace it with:
  - Nom ('NPs' that can be subjects of finite verbs)
  - Acc ('NPs' that can be objects of verbs or prepositions)
  - Neu ('NPs' that can be either)
- Next, we assert the inequalities
  - Neu $\leq$ Nom
  - Neu $\leq$ Acc
- Finally, we revise the lexicon correspondingly (next slide).

Lexicon Revised and Expanded to Analyze Case

(Semantics omitted since it is not relevant.)

$\vdash \text{pedro}; \text{Neu}$

$\vdash \text{chiqita}; \text{Neu}$

$\vdash \text{maria}; \text{Neu}$

$\vdash \text{she}; \text{Nom}$

$\vdash \text{he}; \text{Nom}$

$\vdash \text{him}; \text{Acc}$

$\vdash \text{her}; \text{Acc}$

$\vdash \lambda_s . s \cdot \text{brayed}; \text{Nom} \multimap \text{S}$

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

$\vdash \lambda_{stu} . s \cdot \text{gave} \cdot t \cdot u; \text{Nom} \multimap \text{Acc} \multimap \text{Acc} \multimap \text{S}$

$\vdash \lambda_{st} . s \cdot \text{believed} \cdot t; \text{Nom} \multimap \bar{\text{S}} \multimap \text{S}$

How Neutral 'NPs' Get Case

This derivation uses Derived Rule Schema 1 twice:

$$\dfrac{\dfrac{\vdash \lambda_{st}.s \cdot \mathrm{beat} \cdot t; \mathrm{Nom} \multimap \mathrm{Acc} \multimap \mathrm{S} \qquad \dfrac{\vdash \mathrm{pedro}; \mathrm{Neu}}{\vdash \mathrm{pedro}; \mathrm{Nom}}}{\vdash \lambda t.\mathrm{pedro} \cdot \mathrm{beat} \cdot t; \mathrm{Acc} \multimap \mathrm{S}} \qquad \dfrac{\vdash \mathrm{chiquita}; \mathrm{Neu}}{\vdash \mathrm{chiquita}; \mathrm{Acc}}}{\vdash \mathrm{pedro} \cdot \mathrm{beat} \cdot \mathrm{chiquita}; \mathrm{S}}$$

How Quantified ‘NPs’ Get Case

- Remember that N (common noun) is a basic tecto, e.g.

  $\vdash \mathrm{donkey}; \mathrm{N}$

  $\vdash \mathrm{farmer}; \mathrm{N}$

- Then, since QP’s like *a donkey* or *every farmer* share with names the ability to serve as either subjects or objects, we revise the definition of QP from $(\mathrm{NP} \multimap \mathrm{S}) \multimap \mathrm{S}$ to $(\mathrm{Neu} \multimap \mathrm{S}) \multimap \mathrm{S}$.
- Then we can derive, e.g.

$$\frac{\vdash \lambda_s.\mathrm{a} \cdot s; \mathrm{N} \multimap \mathrm{QP} \qquad \vdash \mathrm{donkey}; \mathrm{N}}{\vdash \mathrm{a} \cdot \mathrm{donkey}; \mathrm{QP}}$$

- To use QPs as subjects or objects, it becomes necessary to show that a QP can ‘become’ an $(\mathrm{Nom} \multimap \mathrm{S}) \multimap \mathrm{S}$ or an $(\mathrm{Acc} \multimap \mathrm{S}) \multimap \mathrm{S}$.
- This is left as an exercise.

Attributive Adjectives

- We distinguish between **attributive** adjectives, which modify nouns, and **predicative** adjectives, which are usually introduced by a copula (form of the auxiliary verb *be*) or other 'linking' verbs (such as *become*).
- Although many adjectives appear both ways, some (such as *asleep*) can only be predicative, while others (such as *former*) can only be attributive.
- Adapting the usual categorial analysis of modifiers, we analyze attributive adjectives as having tectotype $\mathrm{N} \multimap \mathrm{N}$:

  $\vdash \lambda_s.\mathrm{lazy} \cdot s; \mathrm{N} \multimap \mathrm{N}$

  $\vdash \lambda_s.\mathrm{former} \cdot s; \mathrm{N} \multimap \mathrm{N}$
- Then we can analyze common noun phrases like:

$$\frac{\vdash \lambda_s.\mathrm{lazy} \cdot s; \mathrm{N} \multimap \mathrm{N} \qquad \vdash \mathrm{donkey}; \mathrm{N}}{\vdash \mathrm{lazy} \cdot \mathrm{donkey}; \mathrm{N}}$$

Predicative Adjectives

- As a first approximation, we analyze predicative adjectives with a new basic tectotype PrdA:

  $\vdash \mathrm{lazy}; \mathrm{PrdA}$

  $\vdash \mathrm{asleep}; \mathrm{PrdA}$

- We can't do anything with these yet, but we are about to fix that.

Introducing Existential *Be*

- We distinguish between **existential** *be*, as in *there is a donkey*, and **predicational** *be*, as in *Chiquita is lazy*.
- Existential *be* requires a dummy *there* subject and a QP complement which is subject to certain semantic constraints (roughly, it must be indefinite):

  $\vdash \lambda_{st}.s \cdot \text{is} \cdot t; \text{There} \multimap \text{QP} \multimap \text{S}$

- Optionally, existential *be* can take an additional 'predicative' complement, but we postpone consideration of those for the time being.

Introducing Predicational *Be*

- As a first approximation, predicational *be* takes a noun phrase subject, which for finite forms of *be* must be nominative, and a predicative adjective complement (actually, there are several other options for complements of *be*, as we soon will see):

  $\vdash \lambda_{st}.s \cdot \text{is} \cdot t; \text{Nom} \multimap \text{PrdA} \multimap \text{S}$

- But there is a problem. Some PrdAs demand a dummy *it* subject, while most require a 'normal' nondummy subject:

  1. Chiquita/He/She is lazy/asleep.
  2. $*$ Chiquita/He/She is rainy.
  3. It is rainy.
  4. $*$ It is lazy/asleep. (where *it* is not referential)

- How does the copula know what kind of subject its predicative complement expects?

Predicative Adjectives 'Care' about their Subjects

- Even though a predicative adjective cannot directly take a subject, if a copula takes it as a complement, it 'tells' the copula what kind of subject to take.
- We analyze this by treating predicative adjectives **tectogrammatically** (and semantically) as functors, but phenogrammatically as just strings:

  $\vdash \text{rainy}; \text{It} \multimap \text{PrdA}$

  $\vdash \text{obvious} : \bar{\text{S}} \multimap \text{PrdA}$

  $\vdash \text{lazy} : \text{Nom} \multimap \text{PrdA}$

  The 'Nom' in the last entry is not quite right, but it will take some development to see why.
- We will analyze nonfinite verb phrases (infinitivals, base-form verb phrases, and participial phrases) in essentially the same way, but with PrdA replaced by other basic tecto-types (Inf, Bse, Prp, Psp, and Pas).

Predicational *Be*, Take Two

- Now, we replace our old lexical entry for predicational *is*:

  $\vdash \lambda_{st}.s \cdot \text{is} \cdot t; \text{Nom} \multimap \text{PrdA} \multimap \text{S}$

  with the following **schema**:

  $\vdash \lambda_{st}.s \cdot \text{is} \cdot t; A \multimap (A \multimap \text{PrdA}) \multimap \text{S}$

  where A is a metavariable that ranges over tectos.
- This analysis corresponds to what is called **raising to subject (RTS)** in other frameworks.
- In essence, *is* says: ‘I don’t care what my subject is, as long as my complement is happy with it’.
- We can use the same trick to analyze other verbs (and nonverbal predicatives) traditionally analyzed in terms of RTS (e.g. modals and other auxiliaries, *seem*, *tend*, etc.).

Problems with Raising (1/2)

- There are other problems, though: there are some verbs, traditionally called **raising to object (RTO)** verbs, that feel the same way about their object as RTS verbs feel about their subject, for example *considers*:

  1. Pedro considers it rainy.
  2. Pedro considers that Chiquita brays obvious.
  3. Pedro considers Chiquita/her/∗she lazy.

- For such verbs, if the object is a pronoun, it has to be accusative.

Problems with Raising (2/2)

- So if we try to analyze RTO on a par with RTS with a lexical entry like

  $\vdash \lambda_{stu}.s \cdot \text{considers} \cdot t \cdot u; \text{Nom} \multimap A \multimap (A \multimap \text{PrdA}) \multimap \text{S}$

  it will interact badly with the lexical entry

  $\vdash \text{lazy} : \text{Nom} \multimap \text{PrdA}$

  to overgenerate things like

  $*$ Pedro considers she lazy.

  while failing to generate the correct

  Pedro considers her lazy.

Fixing the Undergeneration Problem with Raising (1/2)

- The undergeneration problem arises with RTO because the lexical entries for predicative adjectives like *lazy* (and for nonfinite verbs like *bray*) are demanding Nom subjects.
- This works when the 'unrealized' subject is 'raised' to the subject of a finite verb (such as *is*), but not when it is 'raised' to object, where an **accusative** pronoun is needed.
- An easy fix would be to add a second entry with tecto type Acc $\multimap$ PrdA (and likewise for nonfinite verb forms).
- But we can avoid doubling up all these lexical entries if instead we eliminate all the Nom $\multimap$ PrdA entries and replace them with entries with tectotype PRO $\multimap$ PrdA, where PRO is a new basic tectotype ordered as follows:
  - Nom $<$ PRO
  - Acc $<$ PRO

Fixing the Undergeneration Problem with Raising (2/2)

- Then in the lexicon we need only list

  $\vdash \text{lazy}; \text{PRO} \multimap \text{PrdA}$

- From this we can derive the signs needed as complements to *is* and *considers*, respectively, by Derived Rule Schema 2:

  $\vdash \text{lazy}; \text{Nom} \multimap \text{PrdA}$

  $\vdash \text{lazy}; \text{Acc} \multimap \text{PrdA}$

- Note that while Neu is **overspecified** between Nom and Acc, PRO is **underspecified** between Nom and Acc.
- Cf. Chomsky's PRO, which is supposed to occur in non-case-assigned positions such as subject of infinitive.
- And so (as in HPSG but unlike GB or MP), predicatives and nonfinite VPs don't actually 'have' subjects.

Fixing the Overgeneration Problem with Raising (1/2)

- As it stands, our analysis overgenerates:

  1. $*$ Pedro considers she lazy.
  2. $*$ Her is lazy.

  because the *A*s in the lexical schemas for *is* and *considers* can be instantiated (inter alia) as Nom or Acc.
- *is* doesn't care what its subject is as long as its complement is happy with it, and *considers* doesn't care what its object is as long as its complement is happy with it.
- But *is* should be insisting that if its subject is a (nondummy) NP, then it has to be nominative.
- And *considers* should be insisting that if its object is a (nondummy) NP, then it has to be accusative.
- We'll solve these problems by limiting the possible instantiations of the type variable A in the lexical entries, in different ways.

Fixing the Overgeneration Problem with Raising (2/2)

- We add two new basic tectotypes NOM and ACC.
- NOMs are things that can be subjects of finite RTS verbs.
- ACCs are things that can be objects of RTO verbs.
- Next we add more tectotype inequalities:
  - Nom $<$ NOM
  - It $<$ NOM
  - There $<$ NOM
  - Acc $<$ ACC
  - It $<$ ACC
  - There $<$ ACC
- And finally, we revise the lexical schemas for *is* and *considers* as follows:

  $\vdash \lambda_{st}.s \cdot \text{is} \cdot t; A \multimap (A \multimap \text{PrdA}) \multimap \text{S}\ (A \leq \text{NOM})$

  $\vdash \lambda_{stu}.s \cdot \text{considers} \cdot t \cdot u; \text{Nom} \multimap A \multimap (A \multimap \text{PrdA}) \multimap \text{S}$
  $(A \leq \text{ACC})$

Subjects of Nonfinite Verbs (1/2)

- As we've seen, the type requirement for subjects of nonfinite verbs whose finite counterpart would require a Nom is PRO.
- And the type requirement for subjects of finite RTS verbs is NOM.
- But what is the type requirement for the subject of a nonfinite RTS verb, such as *be* or *to*? It is less constrained than objects of RTO verbs or subjects of finite RTS verbs, because no case requirement is imposed on it.
- We can handle this by positing a new tecto, called NP (because it plays a role analogous to that of NP-trace in GB theory), of which NOM, PRO, and ACC are subtypes.
- Then we write lexical schemas schematized over values of A which are subtypes of NP:

$$\vdash \lambda_s.\mathrm{be} \cdot s; (A \multimap \mathrm{PrdA}) \multimap A \multimap \mathrm{Bse} \ (A \leq \mathrm{NP})$$

$$\vdash \lambda_s.\mathrm{to} \cdot s; (A \multimap \mathrm{Bse}) \multimap A \multimap \mathrm{Inf} \ (A \leq \mathrm{NP})$$

Subjects of Nonfinite Verbs (2/2)

- Notice that in the preceding lexical entries, the tectos are written with the complements as the **intial** arguments and the subject (which cannot be taken directly as an argument) **last**.
- This same practice is followed for all nonfinite verbs (and complement-taking nonverbal predicatives). Compare:

$$\vdash \lambda_{st}.s \cdot \mathrm{beats} \cdot t; \mathrm{Nom} \multimap \mathrm{Acc} \multimap \mathrm{S}$$

$$\vdash \lambda_{s}.\mathrm{beat} \cdot s; \mathrm{Acc} \multimap \mathrm{PRO} \multimap \mathrm{Bse}$$

- Although verbs (other than *to*) don't have infinitive forms, roughly that effect results from syntactic combination:

$$\dfrac{\dfrac{\lambda_s.\mathrm{to} \cdot s; (A \multimap \mathrm{Bse}) \multimap A \multimap \mathrm{Inf}}{\lambda_s.\mathrm{to} \cdot s; (\mathrm{PRO} \multimap \mathrm{Bse}) \multimap \mathrm{PRO} \multimap \mathrm{Inf}} \qquad \vdash \mathrm{bray}; \mathrm{PRO} \multimap \mathrm{Bse}}{\vdash \mathrm{to} \cdot \mathrm{bray}; \mathrm{PRO} \multimap \mathrm{Inf}}$$

Here for expository purposes we pretend that instantiation of a schema is a unary rule (of course it isn't really.)

Introducing Predicatives

- Besides predicative adjectives ($A \multimap \mathrm{PrdA}$), the existential copula that takes an additional complement besides the NP also allows three other kinds of complements: predicative PPs ($A \multimap \mathrm{PrdP}$), present participials ($A \multimap \mathrm{Prp}$), and passive participials ($A \multimap \mathrm{Pas}$).
- And the predicational copula allows all of these, as well as predicative NPs ($A \multimap \mathrm{PrdN}$),
- So we propose two new basic tectotypes PrdnN (non-nominal predicative) and Prd (predicative), and assert: $\mathrm{PrdA} < \mathrm{PrdnN}$, $\mathrm{PrdP} < \mathrm{PrdnN}$, $\mathrm{Prp} < \mathrm{PrdnN}$, $\mathrm{Pas} < \mathrm{PrdnN}$, $\mathrm{PrdN} < \mathrm{Prd}$, and $\mathrm{PrdnN} < \mathrm{Prd}$.
- Then revise the schema for the predicational copula to: $\vdash \lambda_{st}.s \cdot \mathsf{is} \cdot t; A \multimap (A \multimap \mathrm{Prd}) \multimap \mathrm{S}\ (A \leq \mathrm{NOM})$
- And revise the two-complement existential copula to: $\vdash \lambda_{stu}.s \cdot \mathsf{is} \cdot t \cdot u; \mathrm{There} \multimap \mathrm{QP} \multimap (\mathrm{PRO} \multimap \mathrm{PrdnN}) \multimap \mathrm{S}$

Two Problems with Predicatives

- So far we have said nothing about where predicative NPs and PPs come from.
- This leads into the topic of **nonlogical rules**, which we will come back to.

Introducing Nonlogical Rules (1/3)

- The last problem on Problem Set One showed that it is harder to analyze relative clauses (RCs) without relative pronouns or relativizers than RCs with them.
- Why? Because there is no overt expression responsible for converting a gappy sentence into a postnominal modifier.
- The usual solution is to posit an inaudible relative pronoun or relativizer along the following lines:

$$\vdash \lambda_{fs}.s \cdot (f\ \mathbf{e}); (\mathrm{Acc} \multimap \mathrm{S}) \multimap \mathrm{N} \multimap \mathrm{N}; \mathsf{that}$$

  Note: This isn't quite right, since it disallows the relativization of embedded subjects within the RC.
- Such lexical entries are reminiscent of the 'null functional heads' of mainstream generative grammar.

Introducing Nonlogical Rules (2/3)

- Equivalently, we can posit a special-purpose nonschematic inference rule:

$$\frac{\Gamma \vdash f; \mathrm{Acc} \multimap \mathrm{S}; P}{\Gamma \vdash \lambda_s.s \cdot (f\ \mathbf{e}); \mathrm{N} \multimap \mathrm{N}; \lambda_Q.P \text{ that } Q}$$

- Such rules are called **nonlogical** because they are not included in the linear logic-based rule schemas already introduced.
- It turns out that the need for such rules arises often.
- The difference between logical and nonlogical rules seems to correspond closely to the distinction in mainstream generative grammar between 'core grammar' and the 'marked periphery'.

Introducing Nonlogical Rules (3/3)

- Notice that even though the null relativizer rule is nonlogical, its semantics is logical in a certain sense: its reference at any world w is definable without any nonlogical constants except @ itself:

$$\vdash \forall_w.\mathsf{that}@w = \lambda_{PQx}.(P\ x)@w \wedge (Q\ x)@w$$

- As we'll see, nonlogical rules that arise in practice often (but not always) have this property. Why is this?

Where Do Predicative NPs Come From? (1/2)

- In order for sentences like *Chiquita is lazy* to get the right semantics, the predicational copula

$$\vdash \lambda_{st}.s \cdot \text{is} \cdot t; A \multimap (A \multimap \text{Prd}) \multimap \text{S}; \mathsf{prd}\ (A \leq \text{NOM})$$

must predicate its complement's meaning of its subject's meaning.

- And so its own meaning must be the **predication** combinator $\mathsf{prd} =_{\text{def}} \lambda_{xP}.P\ x$.
- But this won't work if the complement is an ordinary NP such as *Burrita*; we need a 'version' of *Burrita* (or any other NP that could occur postcopularly) that has tecto PrdN and a property meaning, in the present case $\lambda_x.x\ \mathsf{exteq}\ \mathsf{b}$, where $\mathsf{exteq} : \text{e} \rightarrow \text{e} \rightarrow \text{p}$ is subject to the meaning postulate:

$$\vdash \forall_{xyw}.(x\ \mathsf{exteq}\ y)@w \leftrightarrow (x@w = y@w)$$

Where Do Predicative NPs Come From? (2/2)

- It will come as no surprise that there are two logically equivalent ways to manage this:
  1. a nonlogical rule:
$$\frac{\Gamma \vdash a; \mathrm{Acc}; b}{\Gamma \vdash a; \mathrm{PRO} \multimap \mathrm{PrdN}; \lambda_x . x \ \mathsf{exteq}\ b}$$
  2. a lexical entry:
$$\vdash \lambda_s . s; \mathrm{Acc} \multimap \mathrm{PRO} \multimap \mathrm{PrdN}; \mathsf{exteq}$$
- Either way we can derive:

$$\vdash \mathrm{chiquita} \cdot \mathrm{is} \cdot \mathrm{burrita}; \mathrm{S}; \mathsf{c\ exteq\ b}$$

Three Kinds of Prepositional Phrases

The term ‘prepositional phrase’ is used for (at least) three different kinds of expressions in English:

1. Pedro depends **on Chiquita**. (semantically vacuous)
2. **On Chiquita** is Pepito’s favorite place to be. (refers to a location (spatiotemporal region))
3. Pepito is **on Chiquita**. (predicates being at a location)

Semantically Vacuous Prepositions

- PPs with specific semantically vacuous prepositions can be subcategorized for by verbs, e.g.

  $\vdash \lambda_s.\text{depend} \cdot s; \text{On} \multimap \text{PRO} \multimap \text{Bse}; \lambda_{yx}.\mathsf{depend}\ x\ y$

- We analyze them as having different tectotypes, e.g. On, By, For, With, etc., with the meaning of the PP determined by the prepositional object:

$$\frac{\vdash \lambda_s.\text{on} \cdot s; \text{Acc} \multimap \text{On}; \lambda_x.x \qquad \vdash \text{chiquita}; \text{Acc}; \mathsf{c}}{\vdash \text{on} \cdot \text{chiquita}; \text{On}; \mathsf{c}}$$

- Is there any reason to consider different semantically vacuous prepositions as subtypes of a common tecto?

Nonpredicative Locative Prepositions

- Some prepositions combine with an Acc to form an expression which refers to, or perhaps existentially quantifies over, (a) certain location(s) associated with the entity denoted by that Acc.
- Let us call such expressions **locatives** (Loc) and such prepositions **nonpredicative locative** prepositions.
- Something to think about: how should Loc fit into our ordering of basic tectos?
- Assuming locations are certain kinds of entities, the meaning of a locative preposition is a function that maps entities to an associated (quantifer over) location(s), so that e.g. ($\mathsf{on}\ \mathsf{c}$) denotes the 'on Chiquita' location.
- So we have lexical entries like:

$$\vdash \lambda_s.\mathrm{on} \cdot s; \mathrm{Acc} \multimap \mathrm{Loc}; \mathsf{on}$$

Prepositions that Predicate Location (1/2)

- Many prepositions, here analyzed with tecto Acc $\multimap$ PrdP, predicate:

  1. This present is **for** you.
  2. This book is **about** bats.
  3. Your argument is **without** merit.

- Among these are ones that predicate location of the subject denotation at a location associated with the denotation of the prepositional object:

  1. Pepito is on Chiquita.
  2. Chiquita is behind Pedro.
  3. Pedro is beside Maria.

- Let's call these **predicative locative prepositions**.

Prepositions that Predicate Location (2/2)

- Clearly the location at which a predicative locative PP locates the subject is the **same** as the one denoted by the corresponding nonpredicative locative PP. E.g.:
- *Pepito is on Chiquita* predicates of Pepito *being at* the location denoted by the nonpredicative PP *on Chiquita*
- We can analyze this correspondence with a nonlogical rule

$$\frac{\Gamma \vdash s; \mathrm{Loc}; l}{\Gamma \vdash s; \mathrm{PRO} \rightarrow \mathrm{PrdP}; \lambda_x.x \ \mathsf{at}\ l}$$

or the equivalent lexical entry:

$\vdash \lambda_s.s; \mathrm{Loc} \multimap \mathrm{PRO} \multimap \mathrm{PrdP}; \lambda_{lx}.x \ \mathsf{at}\ l$

- Be careful not to confuse the locative at with the constant @ denoting the being-true-at relation!
- Note that this rule is nonlogical in a strong, semantic sense, because its meaning contribution involves the nonlogical constant at.

More about Nonlogical Rules

- More examples: rules that turn
  - plural and mass N's into Neu's
  - predicatives into 'absolutive' sentence modifiers
  - nonnominal predicatives into postnominal modifiers (so-called 'reduced relatives')
- Do languages have lots of nonlogical rules, or just a few?
- Are nonlogical rules which are semantically logical the norm or are they exceptional?
- What is the range of possible non-logical meanings for nonlogical rules?

Control (1/5)

- We saw that PRO is used for the unrealized subject of nonfinite verbals and predicatives where the subject ‘plays a semantic role’ (and so dummy subjects are disallowed).
- If such an expression is the complement of
  - a RTO verb, then the PRO is ‘identified with’ the upstairs Acc object (here indicated informally by subscripts)

    [consider her$_{\mathit{1}}$ [PRO$_{\mathit{1}}$ conservative]]
  - a finite RTS verb, then the PRO is ‘identified with’ the upstairs Nom subject

    [he$_{\mathit{1}}$ seems [PRO$_{\mathit{1}}$ conservative]]
  - a nonfinite RTS verb or predicative, then the PRO is ‘identified with’ the upstairs PRO subject.

    [PRO$_{\mathit{1}}$ be [PRO$_{\mathit{1}}$ conservative]]
- In all these cases, the upstairs object or subject identified with the PRO complement subject *plays no semantic role with respect to the upstairs verbal/predicative.*

Control (2/5)

- But expressions with a PRO subject requirement are not always complements of raising verbs. For example, they can themselves be subjects, as in *to err is human*. Here the property of being human is being predicated of another property, the property of erring.
- Such expressions can also be complements of a verb (or predicative), which (in a sense to be made precise) 'identifies' the unrealized downstairs subject *semantically* with one of its own arguments (either the subject or the object) which *does* play a semantic role upstairs.

Control (3/5)

- Examples:

  1. Chiquita tried to sing.
  2. Pedro persuaded Chiquita to sing.

- Verbs like these are often analyzed as describing a relation between one or two entities and a proposition about one of those entities (in the examples above, the proposition about Chiquita that she sings).
- That entity (here, Chiquita), or the corresponding upstairs argument position (subject of *tried* or object of *persuaded*), is said to *control* the PRO subject of the complement.
- In such cases the higher verb is called a **control** verb (and likewise for predicatives).

Control (4/5)

- Control verbs are also called **equi** verbs because in early TG they were analyzed by a transformation ('equi-NP deletion') that deleted the complement subject (which was assumed to be identical with the controller).
- By comparsion, raising verbs in TG were analyzed by a different transformation ('raising') that moved the complement subject to a higher position in the tree.

Control (5/5)

- In LG, the analysis of control makes no tectogrammatical connection between the complement subject and the controller, instead handling the connection semantically:

  $\vdash \lambda_{st}.s \cdot \text{tries} \cdot t; \text{Nom} \multimap (\text{PRO} \multimap \text{Inf}) \multimap \text{S}; \lambda_{xP}.\mathsf{try}\ x\ (P\ x)$

  $\vdash \lambda_{stu}.s \cdot \text{persuaded} \cdot t \cdot u; \text{Nom} \multimap \text{Acc} \multimap (\text{PRO} \multimap \text{Inf}) \multimap \text{S}; \lambda_{xyP}.\mathsf{persuade}\ x\ y\ (P\ y)$

- Alternatively, control verb meanings can be treated as relations between one or two entities and a *property*:

  $\vdash \lambda_{st}.s \cdot \text{tries} \cdot t; \text{Nom} \multimap (\text{PRO} \multimap \text{Inf}) \multimap \text{S}; \lambda_{xP}.\mathsf{try}\ x\ P$

  $\vdash \lambda_{stu}.s \cdot \text{persuaded} \cdot t \cdot u; \text{Nom} \multimap \text{Acc} \multimap (\text{PRO} \multimap \text{Inf}) \multimap \text{S}; \lambda_{xyP}.\mathsf{persuade}\ x\ y\ P$

  with the relationship between the controller the property captured via nonlogical axioms (‘meaning postulates’) of the semantic theory.

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 in the mainstream generative grammar tradition.

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."

GB Theory's 'Empty Categories' (1/2)

- GB (early 1980's) posited four kinds of EC's
  - (little) pro, essentially inuadible definite pronouns, not relevant for the present discussion
  - trace (aka 'syntactic variable')
  - NP-trace
  - (big) PRO
- These last three figured, respectively, in the analysis of:
  - wh-movement, later subsumed under $\bar{\text{A}}$-movement (wh-questions, relative clauses, topicalization, clefts, pseudoclefts, etc.)
  - NP-movement, later called A-movement (passive, raising)
  - control

GB Theory's 'Empty Categories' (2/2)

- The theoretical assunptions about how these three kinds of empty elements worked never seemed to add up to a consistent story about TCs.
- In LG we have counterparts of all three.
- In due course we'll see how LG fares in accounting for TCs.
- First a glance at how the GB EC's were supposed to work.

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

Something moves, 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

NP-Movement/A-Movement

Something moves 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 is 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 $\bar{\text{A}}$-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 incompetent.
  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)

Basic Tectos Involved in Analysis of TCs (1/2)

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

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

For (nonpredicative *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)

Basic Tectos Involved in Analysis of TCs (2/2)

- 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*

Review of Basic Tecto Ordering

Neu < Nom

Neu < Acc

Nom < PRO

Acc < PRO

Nom < NOM

Acc < ACC

It < NOM

It < ACC

NOM < NP

ACC < NP

PRO < NP

PrdA < Prd

Some Nonlogical Constants for Lexical Semantics

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

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

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

$\vdash \mathsf{please} : \mathrm{p}_{\mathit{2}}$
(The first argument is pleasing and the second argument experiences the pleasure.)

$\vdash \mathsf{easy} : \mathrm{e} \to \mathrm{p}_{\mathit{1}} \to \mathrm{p}$
(The first argument is the one who has an easy time of it, and the second argument is the 'piece of cake'.

Lexical Entries

$\vdash$ it; It; $*$ (dummy pronoun *it*)

$\vdash$ john; Neu; j

$\vdash$ mary; Neu; 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

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

Nonpredicative "Prepositional" Phrases

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

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

In the absence of clear empirical supportfor calling nonpredicative For-phrases 'prepositional', we just treat For as a basic tecto.

A Base Phrase

$$\dfrac{\vdash \lambda_t.\mathrm{please} \cdot t; \mathrm{Acc} \multimap \mathrm{PRO} \multimap \mathrm{Bse}; \lambda_{yx}.\mathsf{please}\ x\ y \qquad \dfrac{}{\vdash \mathrm{john}; \mathrm{Acc}; \mathsf{j}}}{\vdash \mathrm{please} \cdot \mathrm{john}; \mathrm{PRO} \multimap \mathrm{Bse}; \lambda_x.\mathsf{please}\ x\ \mathsf{j}}$$

An Infinitive Phrase

[1:]

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

- Here A (two occurrences) in the *to* schema was instantiated as PRO (and B in $P : B \rightarrow \mathrm{p}$ as e). This is legitimate because the schematization is over $A \leq \mathrm{NP}$, and in fact $\mathrm{PRO} \leq \mathrm{NP}$.
- 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

[2]:

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

[3:]

$$\dfrac{\dfrac{[2] \qquad [1]}{\vdash \mathrm{easy} \cdot \mathrm{for} \cdot \mathrm{mary} \cdot \mathrm{to} \cdot \mathrm{please} \cdot \mathrm{john}; \mathrm{It} \multimap \mathrm{PrdA}; \lambda_o.\mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ \mathsf{j})}}{\vdash \mathrm{easy} \cdot \mathrm{for} \cdot \mathrm{mary} \cdot \mathrm{to} \cdot \mathrm{please} \cdot \mathrm{john}; \mathrm{It} \multimap \mathrm{Prd}; \lambda_o.\mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ \mathsf{j})}\ \mathrm{D3}$$

- 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 uses the semantics for Control where the infinitive complement is analyzed as a property rather than a proposition.

It is easy for Mary to please John

[4]:

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

$$\frac{[4] \qquad [3]}{\vdash \text{it} \cdot \text{is} \cdot \text{easy} \cdot \text{for} \cdot \text{mary} \cdot \text{to} \cdot \text{please} \cdot \text{john}; \text{S}; \mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ \mathsf{j})}$$

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

A Gappy Infinitive Phrase

[5]:

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

[6:]

$$\cfrac{\cfrac{\vdash \lambda_t.\text{to} \cdot t; (A \multimap \text{Bse}) \multimap A \multimap \text{Inf}; \lambda_P.P \qquad [5]}{s; \text{Acc}; y \vdash \text{to} \cdot \text{please} \cdot s; \text{PRO} \multimap \text{Inf}; \lambda_x.\mathsf{please}\ x\ y}}{\vdash \lambda_s.\text{to} \cdot \text{please} \cdot s; \text{Acc} \multimap \text{PRO} \multimap \text{Inf}; \lambda_{yx}.\mathsf{please}\ x\ y}\ \text{HP}$$

- 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:

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

- 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

[7:]

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

[8:]

$$\dfrac{\dfrac{\dfrac{[7] \qquad [6]}{\vdash \mathrm{easy} \cdot \mathrm{for} \cdot \mathrm{mary} \cdot \mathrm{to} \cdot \mathrm{please}; \mathrm{PRO} \multimap \mathrm{PrdA}; \lambda_y.\mathsf{easy\ m}\ (\lambda_x.\mathsf{please}\ x\ y)}}{\vdash \mathrm{easy} \cdot \mathrm{for} \cdot \mathrm{mary} \cdot \mathrm{to} \cdot \mathrm{please}; \mathrm{Nom} \multimap \mathrm{PrdA}; \lambda_y.\mathsf{easy\ m}\ (\lambda_x.\mathsf{please}\ x\ y)}\ \mathrm{D2}}{\vdash \mathrm{easy} \cdot \mathrm{for} \cdot \mathrm{mary} \cdot \mathrm{to} \cdot \mathrm{please}; \mathrm{Nom} \multimap \mathrm{Prd}; \lambda_y.\mathsf{easy\ m}\ (\lambda_x.\mathsf{please}\ x\ y)}\ \mathrm{D3}$$

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

John is easy for Mary to please

[9:]

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

$$[9] \quad \dfrac{\overline{\vdash \text{easy} \cdot \text{for} \cdot \text{mary} \cdot \text{to} \cdot \text{please}; \text{Nom} \multimap \text{Prd}; \lambda_y.\mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ y)}}{\vdash \text{John} \cdot \text{is} \cdot \text{easy} \cdot \text{for} \cdot \text{mary} \cdot \text{to} \cdot \text{please}; \text{S}; \mathsf{easy}\ \mathsf{m}\ (\lambda_x.\mathsf{please}\ x\ \mathsf{j})}$$

- 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!
- *Tough*-constructions are unproblematic for LG.