Logical Grammar: Introduction to Linear Grammar

Carl Pollard

Department of Linguistics
Ohio State University

July 11, 2011

LG 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 in the form

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

where

  - $a : A$ is a typed term of a HO theory (the **pheno theory**), called the **pheno term**, or simply the **pheno**
  - B is a formula of a LL (the **tecto** logic) called the **tecto type**, or simply the **tecto**
  - $c : C$ is a typed term of a HO theory (the **semantic** theory), called the **semantic term**, or simply the **semantics**

The Pheno Theory

- There is a basic type s (strings (of phonological words))
- The nonlogical constants are:
  - $\mathbf{e} : \mathrm{s}$, which denotes the **null** string
  - a large number of string constants which denote phenos of syntactic words, such as it, rains, fido, barks, etc.
  - $\cdot : \mathrm{s} \rightarrow \mathrm{s} \rightarrow \mathrm{s}$, which denotes concatenation (written infix)
- The nonlogical axioms are (here $s, t, u : \mathrm{s}$):

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

  These axioms say that the set of strings forms a monoid with concatenation as the associative operation and the null string as the identity element.

The Tecto Logic

This is (implicative intuitionistic propositional) LL, with basic tecto types (i.e. atomic formulas) such as NP, It, S, N, etc.

The inventory of tecto types will be extended and refined as we develop a fragment.

The Semantic Theory

This is the theory of hyperintensional semantics (HS) already introduced, augmented with:

- nonlogical constants, for meanings of syntactic words
- nonlogical axioms (analogous to Montague's **meaning postulates**)

LG Architecture

In its simplest form, an LG consists of:

- Two kinds of **axioms**:
  - **logical** axioms, 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 LF **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 the contexts are actually just finite **sets**.

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

The Trace Axiom Schema

Full form:

$$x : A; B; z : C \vdash x : A; B; z : C$$

Short form (when types of variables are known):

$$x; B; z \vdash x; B; z$$

Two Lexical Entries to Get Started

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

Recall that $*$ is the logical constant of type T!

$\vdash \lambda_s.s \cdot \text{rains}; \text{It} \multimap \text{S}; \lambda_o.\mathsf{rain}$

Here o is of type T, and the constant rain is of type 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

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

Some New Constants for Lexical Semantics

$\vdash$ p : e (Pedro)

$\vdash$ c : e (Chiquita)

$\vdash$ rain : p

$\vdash$ bray : p_1

$\vdash$ donkey : p_1

$\vdash$ farmer : p_1

$\vdash$ own : p_2

$\vdash$ beat : p_2

$\vdash$ give : p_3

$\vdash$ believe : $\mathrm{e} \rightarrow \mathrm{p} \rightarrow \mathrm{p}$

$\vdash$ persuade : $\mathrm{e} \rightarrow \mathrm{e} \rightarrow \mathrm{p} \rightarrow \mathrm{p}$

An LG Proof

Here both axiom instances are lexical entries, and the only rule instance is Modus Ponens.

Unsimplified:

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

Simplified:

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

As in MG, we use TLC term equivalences and meaning postulates 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

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

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

$\vdash \lambda_s.\text{that} \cdot s; \text{S} \multimap \bar{\text{S}}; \lambda_p.p$

(Here $\bar{\text{S}}$ is a new basic tecto for complementized finite clauses.)

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

Note: The finite verb entries are written to facilitate the verb combining first with the subject, then with the complements, in a derivation. This is the reverse of how things are usually done. Reasons for this will be given later.

Another LG Proof

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

Yet Another LG Proof

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

Note that we had to shrink this to tiny to fit it on the slide!
This approach of course has its limits.

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:

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

Of course this approach also has its limits.

An Oversized LG Proof

$$\dfrac{\dfrac{\vdash \lambda_{st}.s \cdot \text{believes} \cdot t; \text{NP} \multimap \bar{\text{S}} \multimap \text{S} \qquad \vdash \text{pedro}; \text{NP}}{\vdash \lambda_t.\text{pedro} \cdot \text{believes} \cdot t; \bar{\text{S}} \multimap \text{S}} \qquad \dfrac{\vdash \lambda_s.\text{that} \cdot s; \text{S} \multimap \bar{\text{S}} \qquad \dfrac{\vdash \lambda_s.s \cdot \text{brays}; \text{N}}{\vdash \text{chi}}}{\vdash \text{that} \cdot \text{chiquita} \cdot \text{brays}; \bar{\text{S}}}}{\vdash \text{pedro} \cdot \text{believes} \cdot \text{that} \cdot \text{chiquita} \cdot \text{brays}; \text{S}}$$

Another Solution to the Space Problem

[1]:

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

[2]:

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

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

Now Let's Do Some Linguistics: English NPs

- To get started, we assumed tectotypes 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 phrase, occur both as subject of verb 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, some of which (*he, she*) cannot be objects, and others of which (*her, him*) cannot be subjects.
  - Only a few verbs, e.g. *be* and *seem*, allow dummy pronoun subjects; and only a few, e.g. *believe*, allow dummy pronoun 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.
- We could add features to LG as has been done for ACG (de Groote and Maarek 2007) using **dependent typing**.
- Here we explore a different approach proposed by Lambek, that uses an order on the basic tectotypes.
- We start by limiting our attention to sentences which contain only 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).

Ordering Basic Tectotypes

- Lambek proposed ordering the basic syntactic types.
- His proposal was in the context of **pregroup grammar**, which is based on classical bilinear logic, but it works for the tectotypes of LG also.
- For example: we would like to say that the type of NPs which can serve as both subjects and objects is a subtype of the type of NPs that can serve as subjects.
- The machinery for subtyping in HOL won't work here.
- Instead, we just assume a larger inventory of basic tectotypes, and impose an order on them by fiat!
- Notationally, we use the symbol $\leq$ for the imposed order on basic tectotypes, and write $A < B$ for $A \leq B$ and $A \neq B$.
- We assert certain inequalities.
- Then we define $\leq$ to be the smallest order on basic tectotypes that includes all the asserted inequalities.

Ordering Basic Tectotypes to Analyze English Case

- First we discard the type NP and replace it with the types Nom (nominative), Acc (accusative), and Neu (neutral):
  - Nom is for NPs that can be nominative.
  - Acc is for NPs that can be accusative.
  - Neu is for NPs that can be either.
- Next, we assert the inequalities
  - Neu $<$ Nom
  - Neu $<$ Acc
- Then we generalize the Trace Axiom Schema as follows: Generalized Trace Axiom Schema:

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

- Finally, we revise the lexicon as described below.

Two Derived LG Schemas

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

- Derived Rule Schema 1

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

- Theorem Schema

$$f; B' \multimap E; g \vdash f; B \multimap E; g$$

- Derived Rule Schema 2

$$\frac{\Gamma \vdash f; B' \multimap E; g}{\Gamma \vdash f; B \multimap E; g}$$

Lexicon Revised for the Analysis of Case

Semantics omitted since it is not relevant.

$\vdash$ pedro; Neu

$\vdash$ chiqita; Neu

$\vdash$ she; Nom

$\vdash$ he; Nom

$\vdash$ him; Acc

$\vdash$ her; Acc

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

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

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

How Neutral NPs Get Case

This derivation uses Derived Rule Schema 1 twice:

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

Determiners and Common Nouns

- As is usual in CG, we take the type N of common nouns to be a basic tectotype.

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

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

- Then, since noun phrases like *a donkey* share with names the ability to serve as either subjects or objects, we analyze determiners as having the tectotype $\text{N} \multimap \text{Neu}$:

  $\vdash \lambda_s.\text{every} \cdot s; \text{N} \multimap \text{Neu}$

  $\vdash \lambda_s.\text{a} \cdot s; \text{N} \multimap \text{Neu}$

- Then we can derive, e.g.

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

- Later when we deal with the semantics of quantificational noun phrases, we'll revise this analysis.

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 CG analysis of modifiers, we analyze attributive adjectives as having tectotype $\mathrm{N} \multimap \mathrm{N}$:

  $\vdash \mathrm{lazy}; \mathrm{N} \multimap \mathrm{N}$

  $\vdash \mathrm{former}; \mathrm{N} \multimap \mathrm{N}$
- 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:

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

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

- 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 noun phrase complement which is subject to certain semantic constraints (roughly, it must be indefinite):

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

- Optionally, existential *be* can take an additional 'predicative' complement. For now, we put on hold exactly what we mean by 'predicative', and pretend that the only predicative expressions are PrdAs:

  $\vdash \lambda_{stu}.s \cdot \text{is} \cdot t \cdot u; \text{There} \multimap \text{Neu} \multimap \text{PrdA} \multimap \text{S}$

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 complement. Continuing to pretend that the only predicatives are PrdAs, we posit:

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

- 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 tectotypes.
- 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.
- But our PRO is just a tectotype: there aren't any signs which have this type.
- 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*? Evidently, there is none! So we can write lexical schemas with unrestricted type variables such as:

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

  $\vdash \lambda_s.\text{to} \cdot s; (A \multimap \text{Bse}) \multimap A \multimap \text{Inf}$
- Notice that in these lexical entries, the tectotypes are written with the complements as the **intial** arguments and the subject (which cannot be taken directly as an argument) **last**, as in HPSG.

Subjects of Nonfinite Verbs (2/2)

- This same practice is followed for all nonfinite verbs (and complement-taking nonverbal predicatives). Compare:

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

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

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

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

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

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.
- The other problem is more straightfoward: now that the predicative copula is looking for predicatives in general, how can it take as complement specific kinds of predicatives such as predicative adjectives?
- To make it concrete, how to we parse *she is lazy*?
- It turns out that the easy way is to first derive a new rule schema, the LG-ification of a derived rule schema of LL called **hypothetical syllogism** (in-class exercise).

Toward the Next Exercise

Up to now we've pretended the only non-dummy NPs are the third-singular ones. But of course this is not the case. The purpose of this exercise is to elaborate the order of basic tectotypes (for things that can be subjects or objects of verbs) to account for the basic facts about English verb agreement and its interaction with case.

To do this exercise, first recall what the intuitive meanings are of the types we have posited so far:

Background for Exercise: Review of NP-Tectotypes

NOM: things that can be subjects of finite RTS verbs (including finite auxiliaries as well as verbs like *seems* and *tends*)

ACC: things that can be objects of RTO verbs (such the version of *believe* that takes an infinitive complement)

Nom: things that can be subjects of finite verbs which disallow dummy subjects (i.e. verbs where the subject has a semantic role)

Acc: things that can be objects of verbs which disallow dummy objects (i,e, verbs where the object has a semantic role)

PRO: the unrealized subject of nonfinite verbs whose finite counterparts would require a Nom subject

There: dummy pronoun *there*, the subject of the existential copula

It: dummy pronoun *it*, the subject (inter alia) of weather predicates

Neu: things which can be either subjects or objects

Background for Exercise: English Verb Inflection

- With exceptions noted below, English verbs have three finite forms (pres-3rdsng, pres-non-3rdsng, and past), all corresponding to result type S), and three nonfinite forms: base-form (Bse), present participle (Prp), past-participle (Psp—here we simplify by ignoring passive participles).
- Modal auxiliaries have just one form, which is finite.
- Following GPSG and HPSG, we analyze infinitive *to* as the only verb which is listed in the lexicon as infinitive.
- The copula is unique in having distinct present forms *am* (1stsng) and *are* (non-3rdsng other than 1stsng).
- The copula is unique in having two past forms *was* (for 1stsng and 3rdsng) and *were* (similar to *are*).
- The tense of finite forms is analyzed in the semantics, and not reflected in the tectotype.
- The kinds of subject a finite verb takes is reflected by the argument type A of the tectotype $A \multimap \ldots \multimap \mathrm{S}$.

The Exercise (1/2)

Elaborate the order of NP-types just enough to account for the basic agreement facts. Elaboration can consist of adding new types, splitting types we already have into two or more types, and asserting new inequalities.
Your analysis should be presented in four parts:

1. List all the NP-types you will use.
2. Assert inequalities (as few as possible: remember you can deduce more inequalities from reflexivity and transitivity).
3. Specify for each of the following words what tectotype it is lexically assigned: the definite pronouns: *I*, *you*, *he*, *she*, *it*, *they*, *we*, *me*, *him*, *her*, *us*, and *them*; the dummies *it* and *there*; the name *Pedro*; and the bare-plural NP *donkeys*, as in *donkeys bray*.

   Group together words which have the same tectotype.

The Exercise (2/2)

4. Specify for each of the following 'argument positions' which of your types is required for it:

   1. subject of modal verb *can*
   2. subject of *am*
   3. subject of *is*
   4. subject of *are*
   5. subject of *was*
   6. subject of base form *bray*
   7. subject of *brays*
   8. subject of finite *bray*
   9. subject of past form *brayed*
   10. subject of *rains*
   11. subject of existential copula
   12. object of RTO verb *believe*
   13. object of *bites*

Hints

- Not all of your types will show up in both 3 and 4.
- No two of the items in part 4 should have the same type.
- Gender is not syntactically relevant; nor are person and number for accusatives.
- To simplify, analyze the kind of English where dummy *there* always has singular agreement even if the postcopular NP is plural.
- Don't be misled by the fact that *you* is semantically plural! Instead, see what words it has the same distribution as.
- I did it with 15 types, only 7 more than we started with (and many fewer than the result of multiplying out combinations of traditional 'features').
- Of course your answer doesn't have to be the same as mine!

Next Up

- Nonlogical Rules
- Quantifier Scope
- Unbounded dependencies