NONLOCAL DEPENDENCIES VIA VARIABLE CONTEXTS

Carl Pollard
Workshop on New Directions in Type-Theoretic Grammar
ESSLLI 2007
Dublin, August 6—10, 2007

These slides are available at:

http://www.ling.ohio-state.edu/~hana/hog/
OVERVIEW OF THIS TALK

• Dimensions along which grammar frameworks differ
• Identify a specific architecture to be explored here
• Introduce the new framework CVG
• Local dependencies (Merge) in CVG
• Formal inspiration: TLC with variable contexts
• Overview of Nonlocal dependencies in CVG
• Extended example: SLASH dependencies (Overt Move) in English
A Parenthetical Remark

a. In order to relate this subject matter more closely to the workshop, I’m going to use the term ‘Curryesque’ for type-theoretic frameworks along the lines that Reinhard Muskens described in his opening talk.

b. I take the following two characteristics to be criterial for identifying such frameworks:
   i. they recognize a distinct level of tectogrammar (or abstract syntax) with its own (Curry-Howard) proof term calculus; and
   ii. they posit interpretive functions that map from tectogrammar to ‘sound’ and ‘meaning’ respectively, which preserve the term constructors and type constructors of the proof-term calculi.

c. Examples:
   i. ACG (de Groote)
   ii. GF (Ranta)
   iii. HOG (Pollard)
   iv. Lambda Grammar (Muskens)
   v. Minimalist Categorial Grammars (Lecomte & Retoré), etc.

d. The CVG framework I am proposing here evolved out of HOG, but as you’ll see, it is distinctly non-Curryesque.
ARCHITECTURAL DIMENSIONS
OF GRAMMAR FRAMEWORKS
(2) Some Dimensions of Grammar Architecture

- How many ‘Levels’?
- Cascaded? Or fractal?
- Syntactocentric? Or relational?
HOW MANY LEVELS?

6
What do we Mean by Levels?

- We mean the number of components in the tuples that the grammar licenses (or generates)
- Here ‘tuples’ might be
  - literally mathematical tuples, e.g. TLG prosodic/semantic pairs, or Curryesque grammar’s pheno/tecto/semantic triples.
  - feature structures, with the top-level attributes as the components, as in HPSG
  - named simultaneous representations, such as the c-structure/f-structure/s-structure of LFG
  - named nonsimultaneous representations, such as GB’s d-structure, s-structure, PF, and LF
- We disregard for now whether a level is supposed to be a ‘structure’ or a formal term interpreted as a structure (but we’ll come back to this)
(4) **Obviously there are at Least Two Levels!**
Everyone needs (at least) one level for each of the two sides of the Saussurean sign:

- a level closely related to sound
- a level closely related to meaning

(5) **A Level Related to Sound**

- GB’s **phonetic form (PF)**
- ‘Curryesque’ grammar’s **phenogrammar**
- HPSG’s **phon** feature
- LFG’s **c-structure** (or their terminal yields anyway)
- TLG’s **prosodic** component
(6)  A Level Related to Meaning

- GB’s logical form (LF)
- HPSG’s CONTENT feature
- LFG’s semantic structure
- the semantic component of TLG or Curryesque grammar
What Else?

• But are there levels other than sound and meaning?
• More specifically, is there a syntactic level?
• TLG (and CCG, etc.) say emphatically no, the things assigned to categories are prosodic/semantic pairs (not triples or quadruples etc.):

  “the term ‘syntax’ we reserve for the mediation of prosodics and semantics by sentential grammar” (Morrill, Fadda, and Valentín 2007)

• A question for TLGists: What about the Curry-Howard proof terms corresponding to categorial derivations, and their interpretation into suitable (say, biclosed monoidal) categories? Why don’t these count as tectogrammatical entities?
A Syntactic Level?

- But other frameworks say yes, there is/are one or more distinct syntactic level(s), so the grammar licenses (at least) triples.
- Examples:
  - GB’s d-structure and s-structure
  - LFG’s f-structure
  - HPSG’s CAT and DTRS features
  - Oehrle’s multidimensional labelled deduction (mid 1990’s)
  - Curryesque frameworks’ abstract syntax/tectostructure
- Note: we can assimilate TLG to ‘Curryesque grammar in a trivial way by calling a prosodic/semantic pair a ‘tectogrammatical object’ and then saying the mappings to prosody and semantics are just the cartesian projections.
- But the real question is whether a syntactic level is needed in a nontrivial sense.
CASCaded? OR FRACTAL?
(9) The Cascaded/Fractal Distinction

- This is a distinction that is relevant only for frameworks that recognize one (or more) syntactic levels.
- The issue is whether
  a. a certain one of the syntactic levels has to be derived for the whole sentence before being “fed” the components of the grammar that determine the sound and meaning levels (cascading), or
  b. all three (or more) levels are present “all the way down” so that even words are triples (fractality)
Examples

Fractal:

– Montague Grammar (with analysis trees as the tectos)
– HPSG
– Curryesque grammars

Cascaded:

– all forms of TG from the ‘Standard Theory’ down through GB theory (d-structure feeds s-structure, which feeds PF and LF)
– Minimalism seems to have been moving toward limited fractality (I think): derivations can have multiple branch points
Curryesque Grammars are Fractal

- Note that Curryesque grammars count as fractal, not as cascaded.

- That is because one does not have to form the tectostructure of the whole sentence before starting to derive the sound and meaning.

- Instead all three levels for a tecto-phrase are determined compositionally from those of its tectogrammatical immediate constituents.

- This does not contradict there being functions from tecto to sound and meaning.

- Type-theoretically, this is because the tecto-to-sound and tecto-to-meaning functions are structure-preserving functors, i.e. they respect the type constructors and term constructors of the tectogrammatical Curry-Howard calculus.
SYNTACTOCENTRIC? OR RELATIONAL?
The Syntactocentric/Relational Distinction

- This distinction is relevant only for frameworks that recognize one or more syntactic levels (for expository simplicity, let’s say just one).

- Let $T$ be the set of all entities (call them tectos) that occur as the tectogrammatical component of some triple licensed by the grammar.

- Then is there a function $\text{phn}$ from tectos to phenos, and a function $\text{sem}$ from tectos to meanings, such that every triple licensed by the grammar is of the form $\langle \text{phn}(t), t, \text{sem}(t) \rangle$ for some syntactic entity $s$?

- That is, are the sound and meaning of a licensed triple uniquely determined by its tecto?
(13) **Syntactocentric vs. Relational Frameworks**

- If yes, the framework is **syntactocentric**. Examples:
  - (Classical) Montague Grammar
  - The Curriesque frameworks (ACG, GF, HOG) etc,
- If no, the framework is **relational**. Examples:
  - Flexible Montague Grammar
  - HPSG
  - LFG
  - Simpler Syntax (Culicover and Jackendoff)
(14) **An Architectural Typology**

To summarize so far, we can group the frameworks mentioned into the following families:

* Cascaded: Just TG.
* No Syntactic Level: Mainstream CG (e.g. TLG, CCG).
* Syntactocentric: Classical MG and Curryesque grammar
* Relational: HPSG, LFG, Flexible MG, Simpler Syntax
Some Observations

- Within the syntactocentric group, the main technical advance of Curryesque grammar over Classical MG is its wholeheartedly type-theoretic tectogrammar, specifically the tecto-level Curry-Howard calculus complete with lambda-binding.

- The Relational group is almost wholly non-type-theoretic. The only exception is Flexible MG, but even Flexible MG (like Classical MG), lacks tecto-level lambda-binding.
I have a Dog in this Fight

- After a long term involvement with a relational framework (HPSG), I started (end of last millennium) to develop a Curryesque one (HOG) to replace it.

- Some reasons:
  - HPSG dealt poorly with some phenomena that CG did better on (such as coordination)
  - HPSG’s type theory was too primitive (no implicative types) to make a clean syntax-semantics interface
  - the formal underpinnings of HPSG were too idiosyncratic (or home-brew) to help build bridges to adjacent disciplines (logic, philosophy, computer science, etc.)
Morrill’s Prophesy

. . . it looks like in the future there will be various joint enterprises in HPSG/CG. People will learn from each other, but it will be more in one direction than the other!
[from 1993 *Ta!* interview]
Back to the Future

Faced with the twin professional challenges of

- wanting to continue to produce detailed linguistic analyses with some potential of being understood by linguistic colleagues, and
- having a usable, teachable framework to for training linguistics students to analyze real phenomena

I came to a renewed appreciation of certain aspects of HPSG.
Why CVG?

- CVG is a sort of course correction from HOG.
- HOG had the advantages shared by type-theoretic frameworks in general, plus lambda-binding in syntax (shared with other Curryesque frameworks).
- But it needlessly gave up HPSG’s main virtue, its relationality.
- The point of CVG is to hold onto the advantage of being type-theoretic (and specifically, having syntactic binding), while restoring HPSG-style relationality.
- So the resulting framework is a kind of type-logical HPSG.
A Wider Perspective

• But much the same architecture as CVG could have been arrived from other points of departure, e.g.
  – by adding syntactic binding to Flexible MG
  – by modifying GB to make it fractal instead of cascaded (i.e. deriving Syntax and LF in parallel instead of feeding the former to the latter.)

• That is why I call it Convergent Grammar: because it can be thought of (depending on where you are coming from) as a repair job (or, I hope, upgrade) of HPSG, or GB, or Montague Grammar, or even of Curryesque grammar.

• Correspondingly, the notation is intended to look like a notational variant of X-grammar to an X-grammariian, for as many values of X as possible.
THE ARCHITECTURE OF
CONVERGENT GRAMMAR (CVC)
CVG has Three Levels

- Prosodics: as in TLG, phonology broadly construed to include word order, external sandhi, phrasing, pitch accents, etc.
- Syntax:
  - Has its own Curry-Howard proof term system (like Curryesque grammar)
  - But it’s multimodal (like TLG), and some of the modes have lambda abstraction (like Curryesque grammar)
  - The syntax has more of an ‘admixture of phenogrammar’ than Curryesque folk are comfortable with (like TLG)—so I’m at pains not to call it ‘tectostructure’ (I’ll doubtless have lapses)
- Semantics: possible-worlds (like all forms of CG), but hyperintensional (like HOG), with propositions as primitives and worlds as ultrafilters.
(22) **CVG is Relational**

- Neither the prosodics-syntax relation $\text{pros}$ nor the syntax-semantics relation $\text{sem}$ is assumed to be a function.
- So, strictly speaking, CVG is not syntactocentric.
- But the relationship between prosodics and semantics is taken to be the *composition* $\text{sem} \circ \text{pros}$ of the syntax-semantics and the prosodics-syntax relations.
(23) **Weak Syntactocentricity**

- This architecture might still be called *weakly* syntactocentric in the following sense:
- Syntax always mediates between prosodics and semantics; there is no *direct* sound-meaning correspondence.
- So our criterion for putting something in syntax is whether there is a prosodics-semantics correlation that it helps account for.
- We say *s mediates* between *p* and *m*, written *med*(p, s, m), provided *p pros s* and *s sem m*.
- So the mediation relation is the set of prosodics/syntax/semantics triples licensed by the grammar.
(24) **Formalizing the Architecture**

We have several different kinds of typing judgments (temporarily ignoring the hypotheses on the left-hand side of the turnstyle):

- $\vdash_p a : A$ asserts that $a$ is a prosodic entity of type $A$
- $\vdash_s a : A$ asserts that $a$ is a syntactic entity of type $A$
- $\vdash_m a : A$ asserts that $a$ is a semantic entity of type $A$
- $\vdash_{\text{pros}} a, b : A, B$ presupposes that $\vdash_p a : A$ and $\vdash_s b : B$; and asserts that $a \text{ pros } b$, i.e. that $a$ and $b$ are connected by the prosody/syntax interface.
- $\vdash_{\text{sem}} a, b : A, B$ presupposes that $\vdash_s a : A$ and $\vdash_m b : B$; and asserts that $a \text{ sem } b$, i.e. that $a$ and $b$ are connected by the syntax/semantics interface.
- $\vdash_{\text{med}} a, b, c : A, B, C$ presupposes that $\vdash_p a : A$, $\vdash_s b : B$, and $\vdash_m c : C$; and asserts that $\text{med}(a, b, c)$, i.e. that $b$ mediates between $a$ and $c$. 

30
(25) A CVG Lexical Entry
\[ \vdash \text{med} /\text{plg}/, \text{pig}, \text{pig}' : \text{Pros}, \text{NP}, \text{Ind} \supset \text{Prop} \]
(26) **Narrowing the Focus**

To keep within time limits, from here on out we will ignore prosodics and focus on the *sem* relation, i.e. the syntax-semantics interface.
A Methodological Observation

- The only purpose of syntax is to mediate between sound and meaning.
- So we will never make a syntactic distinction for the purpose of trying to make $\text{sem}$ be a function (it won’t be).
- Rather, we only reflect into the syntax those semantic distinctions that are correlated with a prosodic distinction.
- We’ll discuss this when we look at nonlocal dependencies.
Local vs. Nonlocal Dependencies

- Like HPSG, CVG distinguishes between
  a. **local** (or **valence**) dependencies, e.g. **SUBJ** (subject), **SPR** (specifier), and **COMP** (complement), and
  b. **nonlocal dependencies**, such as **SLASH** (for extraction), **REL** (for relative pronouns), and **QUE** (for fronted or pied-piped interrogative wh-expressions).

- All are treated as various modes of implication, but only the local ones have Modus Ponens (merge).
LOCAL DEPENDENCIES (MERGE) IN CVG: GRAMMATICAL FUNCTIONS AS MODES OF MODUS PONENS
(29) Some Syntactic Words

⊢ it_{dum} : It
⊢ there_{dum} : There
⊢ that : Fin \rightarrow_{M} \bar{S}
⊢ Fido, Felix, Mary : NP
⊢ rained : It \rightarrow_{SU} Fin
⊢ barked : NP \rightarrow_{SU} Fin
⊢ bit : NP \rightarrow_{C} (NP \rightarrow_{SU} Fin)
⊢ gave : (NP \circ NP) \rightarrow_{C} (NP \rightarrow_{SU} Fin)
⊢ believed_1 : Fin \rightarrow_{C} (NP \rightarrow_{SU} Fin)
⊢ believed_2 : \bar{S} \rightarrow_{C} (NP \rightarrow_{SU} Fin)
⊢ bothered_1 : NP \rightarrow_{C} (\bar{S} \rightarrow_{SU} Fin)
⊢ bothered_2 : (NP \circ \bar{S}) \rightarrow_{C} (It \rightarrow_{SU} Fin)
(30) Some Syntax Rules for Local Dependences

**Fusion:** If $\vdash_s a : A$ and $\vdash_s b : B$
then $\vdash_s a \cdot b : A \circ B$

**Subject Merge:** If $\vdash_s a : A$ and $\vdash_s f : A \rightarrow_{su} B$
then $\vdash_s (\text{su} \ a \ f) : B$

**Specifier Merge:** If $\vdash_s a : A$ and $\vdash_s f : A \rightarrow_{sp} B$
then $\vdash_s (\text{sp} \ a \ f) : B$

**Complement Merge:** If $\vdash_s f : A \rightarrow_{c} B$ and $\vdash_s a : A$
then $\vdash_s (f \ a \ c) : B$

**Marked Merge:** If $\vdash_s f : A \rightarrow_{m} B$ and $\vdash a : A$
then $\vdash (f \ a \ m) : B$
A Notational Sleight of Hand

- Since we are ignoring prosodics, these syntax rules are written with the premises in the left-to-right order in which their prosodic counterparts would occur.
- This means you can figure out what syntactic terms sound like by erasing the punctuation and term constructors, and replacing the words with their prosodies.
- This only works because English has relatively fixed word order and we are ignoring phenomena (such as extraposition and particle movement) that prevent pros from being a function.
A Complex Tectogrammatical Term

\[ (^{SU} \text{it}_{dum} (\text{bothered}_2 (\text{Mary} \cdot (\text{that} (^{SU} \text{Fido barked}^M))^C))) \]

- Terms like these are the Curry-Howard proof terms for the syntax logic.
- These play the same role as traditional labelled bracketings in TG.
- They obviate the need for syntactic proofs to fill pages.
- But we don’t have to write type symbols on the parentheses, since we can figure those out from the knowing the types of the constants.
- Just like in that other, more familiar proof term system, the typed lambda calculus (TLC).
The Original Sin of Syntactic Theory

- This started with TG, but other victims were HPSG and model-theoretic syntax (under the construal that term as acquired in recent years.

- The mistake is thinking that labelled bracketing, or their arboREAL notational variants, are ‘syntactic structures’ whose parts can be moved around and whose ‘configurations’ (dominance, c-command, government, etc.) must be studied,

- Of course they are not; they are Curry-Howard proof terms that denote syntactic entities in a model-theoretic interpretation of the syntactic logic.
More on the Original Sin

- The enormity of the error becomes more evident once we introduce variables and binding operators for nonlocal dependencies.
- The analogous mistake in TLC would be to say that the term \( \lambda_x f(x) \) is formed by the rule of ‘lambda movement’

**d-structure:** \( f(\lambda) \)

**move-\( \lambda \):** the lambda leaves behind a variable \( x \) and then moves to a non-argument position on the left periphery to bind it:

**s-structure:** \( \lambda_x f(x) \)
NONLOCAL DEPENDENCIES IN CVG:
MODES WITHOUT MODUS PONENS
Nonlocal Modes of Implication

- These differ with respect to:
  - whether they have Hypothetical Proof (aka ‘retrieval’ or ‘operator binding’)
  - which structural rules they are subject to
  - whether they are ‘covert’ (operate only on meanings) or ‘overt’ (operate on syntax-semantics pairs)

- The nonlocal implications with Hypothetical Proof (such as $\sim\circ_{\text{SLASH}}$) provide CVG’s chief commonality with Curryesque grammar: lambda abstraction in the syntax.

- This makes respectable Chomsky’s (1970’s) idea of (overt) movement as an inaudible operator that binds a syntactic variable.

- But some nonlocal implications (such as $\sim\circ_{\text{PRON}}$ for unresolved pronouns) have neither Modus Ponens nor Hypothetical Proof.
(36) **Examples where Syntax is not Implicated**

- $\vdash_{\text{sem}} \text{everyone, everyone'} : \text{NP, (Ind } \supset \text{Prop) } \supset \text{Prop}$  
  Note the syntactic category is **not** type-raised.

- The treatment of **quantifier scope** (a natural-deduction reconstruction of Cooper storage) does not impinge on the syntax because quantifier scope doesn’t have a prosodic reflex (e.g. the head verb of the clause where a quantifier takes scope doesn’t have a special inflection).

- Moreover the case of the quantified NP doesn’t have to be remembered at retrieval, just the semantic quantifier and the variable that gets lambda-bound before quantifying in.

- Analogous remarks apply to **wh-in situ construal**, except that a semantic sortal restriction has to be remembered (not a quantifier).
More Examples where Syntax is not Implicated

- **Pronoun resolution** is handled as contraction in the semantics only. Note the antecedent doesn’t have to agree in case with the pronoun.

- **Pied piping:** Nobody is going to check the case of a pied-piped relative or interrogative pronoun at the point where the corresponding semantic variable gets bound.

- Interpretation of **comparative expressions**: the only things about them that has to be remembered are the semantic degree variable and the polarity (‘more’ or ‘less’)

- Interpretation of ‘null specifiers’ in **comparative subdeletion**: the only thing about the subdeleted specifier that has to be has to match ‘remembered’ is its semantic degree variable.
(38) **Nonlocal Dependencies where Syntax IS Implicated**
These are cases where the morphosyntactic features (agreement, case, verb form, governed preposition, etc.) of the dependent element have to be ‘remembered’.

- **Extraction** e.g. the case/agreement of the filler and the gap have to match
- Likewise for **Right Node Raising**—the RNR’ed expression has to match the gap in the first conjunct
- Likewise for associates in **phrasal comparatives**: the associate has to match the complement of *than*. 
FORMAL INSPIRATION:
TLC WITH VARIABLE CONTEXTS
(39) **TLC Terms, the Usual Way**

a. (Hypotheses) If $x$ is a variable of type $A$, then $\vdash x : A$;
b. (Axioms) if $c$ is a constant of type $A$, then $\vdash c : A$;
c. (Modus Ponens) if $\vdash f : A \supset B$ and $\vdash a : A$, then $\vdash f(a) : B$; and
d. (Hypothetical Proof) if $x$ is a variable of type $A$ and $\vdash b : B$, then $\vdash \lambda_x b : A \supset B$.

NB.: Each type has its own set of variables (*Church typing*).
We introduce (variable) contexts in order to make various aspects of hypothesis management explicit.

a. A context is a string of variable/type pairs, written to the left of the turnstile in a typing judgment.

b. The contexts keep track of the undischarged hypotheses.

c. Contexts are strings (not just sets or multisets) because we track
   i. the order of hypotheses
   ii. multiple occurrences of the same hypothesis.

d. We make explicit the structural rules that allow contexts to be restructured.

e. Instead of typed variables, we use a fixed stock of general-purpose variables and let the contexts track what types are assigned to the variables in a given proof (Curry typing).

f. Each variable occurs at most once in a context.

g. We use capital Greek letters as metavariables over contexts.
Implicative TLC Reformulated Using Contexts

Reformulated with contexts, our presentation of TLC looks like this:

a. (Hypotheses) \( x : A \vdash x : A \);

b. (Axioms) \( \vdash c : A \) (\( c \) a constant of type \( A \));

c. (Modus Ponens) if \( \Gamma \vdash f : A \supset B \) and \( \Delta \vdash a : A \), then \( \Gamma, \Delta \vdash f(a) : B \);

d. (Hypothetical Proof) if \( x : A, \Gamma \vdash b : B \), then \( \Gamma \vdash \lambda x . b : A \supset B \);

e. (Weakening) if \( \Gamma \vdash b : B \), then \( x : A, \Gamma \vdash b : B \);

f. (Permutation) if \( x : A, y : B, \Gamma \vdash c : C \), then \( y : B, x : A, \Gamma \vdash c : C \);

g. (Contraction) if \( x : A, y : A, \Gamma \vdash b : B \), then \( x : A, \Gamma \vdash b[y/x] : B \).
Grammar Rules with Variable Contexts

**Fusion:** If $\Gamma \vdash_{\text{sem}} a, c : A, C$ and $\Delta \vdash_{\text{sem}} b, d : B, D$
then $\Gamma; \Delta \vdash_{\text{sem}} a \cdot b, (c, d) : A \circ B, C \land D$

**Subject Merge:** If $\Gamma \vdash_{\text{sem}} a, c : A, C$
and $\Delta \vdash_{\text{sem}} f, v : A \rightarrow_{\text{su}} B, C \supset D$
then $\Gamma; \Delta \vdash_{\text{sem}} (^{\text{su}} a f), v(c) : B, D$
THE SLASH MODE:
MOVEMENT IN ENGLISH
a. BAGELS_i, Kim likes t_i and Sandy hates t_i.
b. *BAGELS_i, Kim likes muffins and Sandy hates t_i.
c. *BAGELS_i, Kim likes t_i and Sandy hates muffins.
(44) Additional Rules for SLASH

**Trace:** $\text{sl}: t, x \vdash_{\text{sem}} t, x : A, B (B \neq T)$

**Finite Move:** If $t, x : A, B; \Gamma \vdash_{\text{sem}} s, p : \text{Fin}, \text{Prop}$
then $\Gamma \vdash_{\text{sem}} \lambda_{t}^{\text{sl}} s, \lambda_{x} p : A \to_{\text{sl}} \text{Fin}, B \supset \text{Prop}$

**Infinitive Move:** If $t, x : A, B; \Gamma \vdash_{\text{sem}} f, v : C \to_{\text{SU}} \text{Inf}, D \supset \text{Prop}$
then $\Gamma \vdash_{\text{sem}} \lambda_{t}^{\text{sl}} f, \lambda_{x} v : A \to_{\text{sl}} (C \to_{\text{SU}} \text{Inf}), B \supset (D \supset \text{Prop})$

**Contraction:** If $\Gamma; t, x : A, B; t', y : A, B; \Delta \vdash_{\text{sem}} c, d : C, D$
then $\Gamma; t, x : A, B; \Delta \vdash_{\text{sem}} c[t'/t], d[y/x] : C, D$

**Topicalization:** If $\Gamma \vdash_{\text{sem}} a, b : A, B$
and $\Delta \vdash_{\text{sem}} c, d : A \to_{\text{sl}} \text{Fin}, B \supset \text{Prop}$
then $\Gamma, \Delta \vdash_{\text{sem}} \tau(a, c), d(b) : \text{Top, Prop}$
ANALYSES
TOPICALIZATION
(45)  a. Felix, Fido bit.
      b. $\vdash_{\text{sem}} \tau(Felix, \lambda^\text{SL}_t (\text{su} \ Fido_n (\text{bit } t^c))), \text{bite}'(\text{Fido}', \text{Felix}') : \text{Top, Prop}$
TOUGH-MOVEMENT
More Lexical Entries

⊢ is, \( \lambda v. (NP \rightarrow_{SU} S_{adj}) \rightarrow_{c} (NP \rightarrow_{SU} Fin), (Ind \supset Prop) \supset (Ind \supset Prop) \)

⊢ easy, \( \lambda r. \lambda x. easy'(r(x)) : (NP \rightarrow_{SL} (NP \rightarrow_{SU} Inf)) \rightarrow_{c} (NP \rightarrow_{SU} S_{adj}), (Ind \supset (Ind \supset Prop)) \supset (Ind \supset Prop) \)

⊢ to, \( \lambda v. (NP \rightarrow_{SU} B_{se}) \rightarrow_{c} (NP \rightarrow_{SU} Inf), (Ind \supset Prop) \supset (Ind \supset Prop) \)
(47) a. Felix is easy to bite.

b. \[ \vdash_{\text{sem}} (su \text{ Felix}_n \ (\text{is} \ \lambda_i^{st} (\text{to} \ (\text{bite} \ t \ c) \ c) \ c)), \]
\[ \text{easy}'(\lambda_z \text{bite}'(z, \text{Felix}')) : \text{Fin}, \text{Prop} \]
EXTRACTION AND STRUCTURAL RULES
(48) Parasitic Gaps

a. This book is hard impossible to start \textit{t} without finishing \textit{t}.
b. These are the reports we filed \textit{t} without reading \textit{t}.
c. Which rebel leader did rivals of \textit{t} assassinate \textit{t}?
(49) **The Prohibition on ‘Movement from Nowhere’**

b. *Who did you see Kim?
c. *The student who I talked to Sandy just arrived.
d. *What this country needs a good five-cent cigar is a good 20-cent draft beer.
e. *It’s Kim that Sandy likes Dana.
f. *John is easy to please Mary.
(50) The Prohibition on Crossed Dependencies

a. i. This violin, even the most difficult sonatas are easy $\lambda_{ti}$ [to play $t_i$ on $t_j$].
   ii. *This sonata, even the most exquisitely crafted violins are difficult $\lambda_{t_j}$ [to play $t_i$ on $t_j$].

b. i. Which problems don’t you know who [to talk to $t_j$ about $t_i$]?
   ii. *Which people don’t you know what, [to talk to $t_j$ about $t_i$]?
(51) Violins and Sonatas

a. My violin, your sonata is easy to play on.

b. \( \vdash_{\text{sem}} \tau(\text{sp \ my violin}, \lambda^\text{nt}_t(\text{sp \ your sonata}) \\ \text{(is \ easy} \ \lambda^\text{nt}_t \ (\text{to \ play \ (t' \ o \ (on \ t \ c)) \ c) \ c) \ c)), \\ \text{easy'}(\text{play-on'}(\text{your'}(\text{sonata'}), \text{my'}(\text{violin'})))) : \text{Fin, Prop} \)