AN INTRODUCTION TO CONVERGENT GRAMMAR

Carl Pollard
INRIA-Lorraine and Ohio State University

Séminaire Calligramme
Nancy, June 3 and June 10, 2008

1 Why yet another New Framework?

(1) **Where CVG Came From, Part 1**

- I worked within the framework of head-driven phrase structure grammar (HPSG) for 15 years (1983-1998).
- Gradually I realized that the purely model-theoretic character of HPSG made it hard to take advantage of advances in linguistic analysis taking place in the categorial grammar (CG) community.
- From 1999 until Spring 2007, I developed higher order grammar (HOG), a **purely derivational** reformulation of HPSG.
- By 'purely derivational' I mean that the derivations were **proofs** as in CG, unlike derivations in transformational grammar (TG), which construct arboreal representations by successive merging and moving.
- HOG turned out to be similar to other **Curryesque** forms of CG devloped by Ranta (GF), de Groote (ACG), Muskens (λG), Anoun and Lecomte (LGL).

(2) **Curryesque Architecture**

In **Curryesque** frameworks:

- There are three kinds of (candidate) linguistic entities, specified by independent components of the grammar:

  syntactic entities (also called **tecto** or **abstract syntax**)
  phonological entities (also called **pheno** or **concrete syntax**)
  semantic entities

- However, syntax still *determines* phonology and semantics, in the sense that the syntax-phonology and syntax-semantics interfaces are both functions from syntax.

(3) **More about Curryesque Architecture**

- HPSG is not Curryesque because the interfaces are not functions.
- TG is not Curryesque because phonological and semantic entities (PF and LF) are obtained by sequences of structural operations that transform syntax.
- Traditional CG (Lambek, Steedman, Moortgat, Morrill, etc.):
  - is not curryesque because there are no tectos.
  - Instead, the syntax logic is just a device for assigning category symbols to pairs of pheno/tecto pairs.
  - Thus traditional CGians never write syntactic Curry-Howard proof terms: their categories are not the kind of things that have inhabitants.
  - What they call "Curry-Howard" terms aren't really: they denote semantic entities, not syntactic ones.

(4) **Where CVG Came From, Part 2**

- About a year ago, I began working on so-called **covert movement** phenomena (scoping of in-situ semantic operators such as quantifiers, interrogative pronouns, comparative operators, etc.).
- That made me realize that I had needlessly given up an important advantage that HPSG had over CG: its **weakly syntactocentric**, **parallel** architecture. (I will explain these terms in a moment.)
- CVG was a course correction from HOG, in order to reincorporate weakly syntactocentric parallelism.

2 Overall Architecture of CVG

(5) **Cascaded vs. Parallel Architecture**

- GB was **cascaded** in the sense that PFand LF are derived from the 'branch point' SS.
- MP is more flexible than GB, allowing multiple branch points.
- But most non-TG frameworks are **parallel**: none of the components is 'derived' from one of the others.
- Instead, independent subsystems generate candidate syntactic and phonological (and in Curryesque frameworks, also syntactic) entities, and then the grammar tells which pairs (or triples) are in the language.

(6) **Strong Syntactocentrism**

In **strongly syntactocentric** frameworks, "syntax feeds semantics" (and sometimes phonology too). Examples:

- In TG, LF (and likewise PF is transformationally derived from syntax at one or more branch points.
- In Lambek's (1988) categorical grammar, semantic interpretation is a structure-preserving functor from a biclosed monoidal category to a topos.
- In Curryesque frameworks, there are functions from syntax to semantics and to phonology.

(7) **Weak Syntactocentrism**

In **weakly syntactocentric** frameworks:

- The syntax-semantics interface determines a **relation** (but not a function) from syntax to semantics;
- The syntax-phonology interface determines a **relation** (but not a function) from syntax to phonology; and
- There is no direct connection between phonology and semantics.
- Examples: LFG (I think), HPSG, Simpler Syntax, CVG

(8) **CVG Compared with ACG and HPSG**

- Purely derivational (like ACG)
- Weakly syntactocentric (like HPSG)
- Parallel (like both)

(9) **Syntax, Semantics, and their Interface in CVG**

- Candidate syntactic derivations are specified by a **syntactic logic** broadly similar to ones used in CG.
- Candidate semantic derivations are specified by a **semantic logic**—RC—closely related to TLC.
- The interface specifies which pairs go together.

(10) **Lecomte 2005**

- The version of **categorial minimalism** (CM) described by Lecomte (2005) has a similar organization.
- CM seeks a categorial embodiment of Chomsky's minimalist ideas.
- By contrast the roots of CVG lie in mid-to-late 1970s nontransformational linguistic theory (e.g. Bach, Cooper, Gazdar, Pullum).

(11) **Observations about Syntactic Proof Terms**

- From the point of view of mainstream CG (CCG,TLG), the weirdest thing about CVG (and ACG) is that we write syntactic proof terms.
- But the terms actually look very much like 1970's TG labelled bracketings with syntactic variables (traces) bound by (often inaudible) operators.
- On the other hand, the fact that (still) Chomsky thinks binding operators literally move seems to show that at some fundamental level he and his followers are not really thinking of syntactic analysis logically; instead the logical terminology is a kind of metaphor.

(12) **Observations about the Interface**

- From the point of view of ACG, the weirdest thing about CVG is that the syntax-semantics interface is not a function.
- But that does not become evident until one looks at semantic ambiguities that, according to CVG, are not reflected in the syntax (such as scope of in situ operators or antecedents of pronouns).

(13) **Signs that an Ambiguity is Purely Semantic**

- No agreement of categorial features, e.g. a pronoun need not agree with its antecedent for case.
- Where the operator can be retrieved does not depend on the syntactic category of the retrieval node, e.g. the scope of a quantifier does not have to be an S (but it does have to be a proposition).
- There is no checking of categorial features associated with the operator, e.g. when a QNP or in situ wh-expression is scoped, its case is not checked.

- In 'covert movement' there is no morphosyntactic marking of the 'extraction path' (e.g. on complementizers or verbs intervening between the trace and the 'landing site') as is the case with 'overt movement' in many languages (Levine and Hukari 2006).

3 The CVG Syntactic Logic

(14) **Format for CVG Syntactic Typing Judgments**

$\vdash a : A$

read 'the (syntactic) term a is assigned the category A.'

N.B. Later; to handle 'over movement', we will add a **context** to the left of the turnstile to track unbound traces.

(15) **CVG Categories**

a. There are some **basic** categories.

b. If A and B are categories, then so are $A \multimap_{\mathrm{F}} B$, where F belongs to a finite set F of **grammatical function names**; these are called **functional** categories with **argument** category A and **result** category B.

(16) **Basic Categories**

We start off with the familiar S and NP, ignoring case, agreement, verb inflection, etc. Others will be added as needed.

(17) **Functional Categories**

a. We start off with the grammatical function names S (subject) and C (complement).[1] Others will be added as needed.

b. Readers who prefer only one flavor of implication in the syntax can ignore the grammatical function subscripts.

c. Lambekians can replace $A \multimap_{\mathrm{S}} B$ by $A \backslash B$, and $A \multimap_{\mathrm{C}} B$ by B / A,

(18) **CVG Syntactic Terms**

a. There are finitely many **(syntactic) words** of each category.

b. There are **syntactic functional terms** of the forms $(f\ a\ ^{\mathrm{F}})$ and $(^{\mathrm{F}}\ f\ a)$

[1] Here CVG betrays its HPSG pedigree.

(19) **(Syntactic) Words**

a. These correspond not just to Bloomfield's "minimal free forms", but also to minimal syntactic units realized phonologically as phrasal affixes, sentence particles, argument clitics, etc.

b. Some of these might be realized nonconcatenatively, e.g. by pitch accents, (partial) reduplication, phonological zero (inaudibility), etc.

(20) **Syntactic Functional Terms**

a. In principle these could always be written $(f\ a\ ^{\mathrm{F}})$, but we write $(f\ a\ ^{\mathrm{C}})$ and $(^{\mathrm{S}}\ a\ f)$ as a mnemonic that in English subjects are to the left and complements to the right.

b. This enables us to read the word order off the syntactic terms, as in EST/GB labelled bracketings.

(21) **Syntactic Schema W (Words)**

$\vdash w : A$ (w a syntactic word of category A)

(22) **Syntactic Schema $\mathbf{M_s}$ (Subject Modus Ponens)**

If $\vdash a : A$ and $\vdash f : A \multimap_{\mathrm{S}} B$, then $\vdash (^{\mathrm{S}}\ a\ f) : B$

(23) **Syntactic Schema $\mathbf{M_c}$ (Complement Modus Ponens)**

If $\vdash f : A \multimap_{\mathrm{C}} B$ and $\vdash a : A$, then $\vdash (f\ a\ ^{\mathrm{C}}) : B$

4 The CVG Semantic Logic: RC

(24) **Cooper Storage and Retrieval (Intuitively)**

a. Suppose that while semantically interpreting a syntactic derivation bottom up, you encounter an in-situ operator with semantics $a : (A \to B) \to C$. Then (**storage**) you can replace a by a variable $x : A$ and place the pair (a, x) in the store.

b. Stores 'percolate upward' through derivations.

c. If you get to a node in the derivation with semantics $b : B$, then (**retrieval**) you can remove (a, x) from the store and replace $b : B$ by $a(\lambda_x b) : C$.

d. In case the in-situ operator is a QNP, $A = \iota$ (individual concept), and $B = C = \pi$ (proposition).

(25) **Sociology of Cooper Storage and Retrieval**

a. Cooper didn't describe it this way. He expressed it all in terms of Montagovian model theory.

b. He was at pains to avoid using λ-calculus, to make it clear he was not advocating some form of LF.

c. Even though this technology makes syntax much simpler (e.g. QNPs are just NPs), the resulting nonfunctionality of semantic interpretation was widely viewed with alarm.

d. However it was embraced in some quarters (e.g. Bach and Partee 1980, and in HPSG).

e. Categorial grammarians tend to describe this technology as 'non-compositional', 'baroque', 'ad hoc', 'a mess', 'as bad as covert movement', etc. (some of the more charitable characterizations).

f. The various schemes of continuized semantics proposed in recent years are presented as major improvements on Cooper storage and retrieval.

(26) **Why RC Calculus?**

a. I still think Cooper's basic approach to scoping in situ operators is the best one.

b. But it suffered from problems of presentation.

c. RC calculus is designed to overcome these presentational problems.

(27) **Toward RC Calculus**

a. RC is a term calculus in the labelled Gentzen-sequent style of natural deduction.

b. As far as I have been able to discern so far, RC can do all the linguistic work that has been done so far with continuized semantics.

c. RC can be directly semantically interpreted, but the easiest way is to transform RC meaning terms into TLC.

d. This transform is the RC analog of the CPS transforms used in "continuized" approaches to semantic interpretation, but much simpler.

e. Morover RC meaning terms look familiar to people used to LF.

(28) **Format for RC Typing Judgments**

$\vdash a : A \dashv \Delta$

a. Read 'the term a is assigned the type A in the **co-context** Δ.'

b. The symbol '$\dashv$' is called the **co-turnstile**.

c. The 'co-' here is mnemonic for 'Cooper', 'covert movement', 'continuation' (since the operators stored in the co-context will scope over their own continuations), and 'Commitment' (to be explained presently).

d. Later, we will add a more familiar-looking **context** to the left of the turnstile, to handle the semantic variables associated with syntactic variables (traces).

(29) **RC Semantic Types**

a. There are some **basic** semantic types.

b. If A and B are types, then $A \rightarrow B$ is a **functional** semantic type with **argument** type A and **result** type B.

c. If A, B, and C are types, then $\mathrm{O}[A, B, C]$, usually abbreviated (following Shan 2004) to A^C_B, is an **operator** semantic type with **binding** type A, **scope** type B, and **result** type C.

(30) **Basic Semantic Types**

For present purposes, we use three basic semantic types:

ι (individual concepts), π (propositions), and κ (polar questions).

Here κ is mnemonic for 'Karttunen' because its transform (see (40 below) into Ty2 will be the Karttunen type for questions.

(31) **Functional Semantic Types**

We employ the following abbreviations for (necessarily curried) functional types:

a. Where σ ranges over strings of types and ϵ is the null string:

i. $A_\epsilon =_{\text{def}} A$

ii. $A_{B\sigma} =_{\text{def}} B \rightarrow A_\sigma$ (e.g. $\pi_{\iota\iota} = \iota \rightarrow \iota \rightarrow \pi$)

b. For $n \in \omega$, $\kappa_n =_{\text{def}} \kappa_\sigma$ where σ is the string of ι's of length n.
For n-ary consituent questions where the constituents questioned all have type ι. E.g. *who likes what* will get type κ_2.

(32) **Operator Types**

a. These will be the semantic types for expressions which would be analyzed in TG as undergoing Ā-movement (either overt or covert).

b. The O-constructor is like Moortgat's (1996) q-constructor, but it belongs to the *semantic* logic, *not* the syntactic one.

c. Thus, for example, while for Moortgat (1996) a QNP would have category q[NP, S, S] and semantic type $(\iota \rightarrow \pi) \rightarrow \pi$, for us it has category (simply) NP and semantic type ι^{π}_{π}.[2].

(33) **RC Semantic Terms**

a. There is a denumerable infinity of **variables** of each type.

b. There are finitely many **basic constants** of each type.

c. There are **functional** terms of the form $(f\ a)$, where f and a are semantic terms.

d. There are **binding** terms of the form $(a_x b)$ where a and b are semantic terms and x is a semantic variable.

e. But there is no λ!

(34) **What goes into the Cooper Store?**

a. The Cooper stores (co-contexts) will contain semantic operators to be scoped, each paired with the variable that it will eventually bind.

b. We call such stored pairs **commitments**, and write them in the form a_x, where the type of x is the binding type of a.

c. Then we call x a **committed** variable, and say that a is **committed** to bind x.

(35) **Semantic Schema A (Nonlogical Axioms)**

$\vdash c : A \dashv$ (c a basic semantic constant of type A)

The basic constants notate meanings of syntactic words (cf. 19).

(36) **Semantic Schema M (Modus Ponens)**

If $\vdash f : A \rightarrow B \dashv \Delta$ and $\vdash a : A \dashv \Delta'$, then $\vdash (f\ a) : B \dashv \Delta; \Delta'$

a. This is the usual ND Modus Ponens, except that co-contexts have to propagated from premisses to conclusions (cf. 24b).

b. Semicolons in co-contexts represent set union (necessarily disjoint, since variables are always posited fresh).

[2]Actually QNPs have to be polymorphically typed. See Pollard 2008a fn. 4.

(37) **Semantic Schema C (Commitment)**

If $\vdash a : A^C_B \dashv \Delta$ then $\vdash x : A \dashv a_x : A^C_B; \Delta$ (x fresh)

a. This is a straightforward ND formulation of Cooper storage (cf. 24a).

b. It generalizes Carpenter's (1997) Introduction rule for Moortgat's $\Uparrow$ (essentially the special case of q where the scope type and the result type are the same), but **in the semantics, not in the syntax**.

(38) **More about Schema C**

a. The syntax-semantics interface will guarantee that when an operator gets committed in the semantics, **no corresponding syntactic change takes place**.

b. This is one of two reasons why the relation between syntax and semantics is **not** a function (the key difference between CVG and other kinds of CG).

c. In this respect, our proposed architecture for the syntax-semantics interface resembles Cooper 1975/1983, Hendriks 1993, and HPSG, as well as most of the continuized-semantics proposals.

(39) **Schema R (Responsibility)**

If $\Gamma \vdash b : B \dashv a_x : A^C_B; \Delta$ then $\Gamma \vdash (a_x b) : C \dashv \Delta$ (x free in b but not in Δ)

a. This is a straightforward ND formulation of Cooper retrieval (24c).

b. It generalizes Carpenter's (1997) Elimination rule for Moortgat's $\Uparrow$, but, again, in the semantics, **not** in the syntax.

c. It is called Responsibility because it is about fulfilling commitments.

d. As with Commitment, the syntax-semantics interface ensures that instances of Responsibility correspond to no syntactic change.

e. This is the other reason why the relation between syntax and semantics is **not** a function.

(40) **The Transform τ from RC Types to Ty2 Meaning Types**

a. $\tau(\iota) = \mathrm{s} \rightarrow \mathrm{e}$

b. $\tau(\pi) = \mathrm{s} \rightarrow \mathrm{t}$

c. $\tau(\kappa) = \tau(\pi) \rightarrow \tau(\pi)$

d. $\tau(A \rightarrow B) = \tau(A) \rightarrow \tau(B)$

e. $\tau(A^C_B) = (\tau(A) \rightarrow \tau(B)) \rightarrow \tau(C)$

(41) **The Transform τ on Terms**

a. Variables and basic constants are unchanged except for their types.

b. $\tau((f\ a)) = \tau(f)(\tau(a))$
The change in the parenthesization has no theoretical significance. It just enables one to tell at a glance whether the term belongs to RC or to Ty2, e.g. (walk' Kim') vs. walk'(Kim').

c. $\tau((a_x b)) = \tau(a)(\lambda_x \tau(b))$
This is the important clause. It says that operator binding consists of abstraction immediately followed by application.

(42) **Ty2 Meaning Postulates for Generalized Quantifiers**[3]

$\vdash$ every' $= \lambda_Q \lambda_P \lambda_w \forall_x (Q(x)(w) \rightarrow P(x)(w))$
$\vdash$ some' $= \lambda_Q \lambda_P \lambda_w \exists_x (Q(x)(w) \wedge P(x)(w))$
$\vdash$ everyone' $=$ every'(person')
$\vdash$ someone' $=$ some'(person')

5 The CVG Syntax-Semantics Interface

(43) **Interface Schema L (Lexicon)**

$\vdash w, c : A, B \dashv$ (for certain pairs $\langle w, c \rangle$ where w is a word of category A and c is a basic constant of type B)

(44) **Interface Schema $\mathbf{M_s}$ (Subject Modus Ponens)**

If $\vdash a, c : A, C \dashv \Delta$ and $\vdash f, v : A \multimap_{\mathrm{S}} B, C \rightarrow D \dashv \Delta'$
then $\vdash ({}^{\mathrm{S}}\ a\ f), (v\ c) : B, D \dashv \Delta; \Delta'$

(45) **Interface Schema $\mathbf{M_c}$ (Complement Modus Ponens)**

If $\vdash f, v : A \multimap_{\mathrm{C}} B, C \rightarrow D \dashv \Delta$ and $\vdash a, c : A, C \dashv \Delta'$
then $\vdash (f\ a\ {}^{\mathrm{C}}), (v\ c) : B, D \dashv \Delta; \Delta'$

(46) **Interface Schema C (Commitment)**

If $\vdash a, b : A, B^D_C \dashv \Delta$, then $\vdash a, x : A, B \dashv b_x : B^D_c; \Delta$ (x fresh)

(47) **Interface Schema R (Responsibility)**

If $\vdash e, c : E, C \dashv b_x : B^D_C; \Delta$ then $\vdash e, (b_x c) : E, D \dashv \Delta$
(x free in c but not in Δ)

[3]Types for Ty2 variables are as follows: $x, y, z : \mathrm{s} \rightarrow \mathrm{e}$ (individual concepts); $p, q : \mathrm{s} \rightarrow \mathrm{t}$ (propositions); $w : \mathrm{s}$ (worlds); and $P, Q : ((\mathrm{s} \rightarrow \mathrm{e}) \rightarrow (\mathrm{s} \rightarrow \mathrm{t}))$ (properties of individual concepts).

6 Analysis of Quantifier Raising in English

(48) **Lexicon for English Fragment**

$\vdash$ Chris, Chris' : NP, ι $\dashv$ (likewise other names)

$\vdash$ everyone, everyone' : NP, ι^{π}_{π} $\dashv$

$\vdash$ someone, someone' : NP, ι^{π}_{π} $\dashv$

$\vdash$ likes, like' : (NP $-\!\circ_{\mathrm{C}}$ (NP $-\!\circ_{\mathrm{S}}$ S), $\iota \to (\iota \to \pi)$ $\dashv$

$\vdash$ thinks, think' : S $-\!\circ_{\mathrm{C}}$ (NP $-\!\circ_{\mathrm{S}}$ S), $\pi \to (\iota \to \pi)$ $\dashv$

(49) **A Simple Sentence**

a. Chris thinks Kim likes Dana.

b. $\vdash$ ($^{\mathrm{S}}$ Chris (thinks ($^{\mathrm{S}}$ Kim (likes Dana $^{\mathrm{C}}$) $^{\mathrm{C}}$))) :
((think' ((like' Dana') Kim')) Chris') : S, π $\dashv$

c. Ty2: think'(like'(Dana')(Kim'))(Chris')

(50) **Quantifier Scope Ambiguity**

a. Chris thinks Kim likes everyone.

b. Syntax (both):
($^{\mathrm{S}}$ Chris (thinks ($^{\mathrm{S}}$ Kim (likes everyone $^{\mathrm{C}}$) $^{\mathrm{C}}$))) : S

c. Semantics (scoped to lower clause):
RC: ((think' (everyone'$_x$((like' x) Kim'))) Chris') : π
Ty2: think'($\lambda_w(\forall_x$(person'$(x)(w) \to$ like'(x)(Kim')$(w)))$)(Chris') : s $\to$ t

d. Semantics (scoped to upper clause):
RC: (everyone'$_x$((think' ((like' x) Kim')) Chris')) : π
Ty2: $\lambda_w(\forall_x$(person'$(x)(w) \to$ think'(like'(x)(Kim'))(Chris')$(w)))$: s $\to$ t

(51) **Raising of Two Quantifiers to Same Clause**

a. Everyone likes someone.

b. Syntax (both): ($^{\mathrm{S}}$ everyone (likes someone $^{\mathrm{C}}$) $^{\mathrm{C}}$) : S

c. $\forall\exists$-reading (RC): (everyone'$_x$(someone'$_y$((like' y) x))) : π

d. $\exists\forall$-reading (RC): (someone'$_y$(everyone'$_x$((like' y) x))) : π

e. These are possible because for generalized quantifiers, the result type is the same as the scope type.

f. Things are not so straightforward in the case of multiple in-situ wh-operators, as we will see in the next talk.

7 Background for the Analysis of Interrogatives

(52) **Ty2 Meaning Types**

a. $\mathrm{s} \rightarrow \mathrm{e}$ (individual concepts) is a Ty2 meaning type.

b. $\mathrm{s} \rightarrow \mathrm{t}$ (propositions) is a Ty2 meaning type.

c. If A and B are Ty2 meaning types, then so is $A \rightarrow B$.

(53) **Extensional Types Corresponding to Ty2 Meaning Types**

These are defined as follows:

a. $\mathsf{E}(\mathrm{s} \rightarrow \mathrm{e}) = \mathrm{e}$

b. $\mathsf{E}(\mathrm{s} \rightarrow \mathrm{t}) = \mathrm{t}$

c. $\mathsf{E}(A \rightarrow B) = (A \rightarrow \mathsf{E}(B))$

(54) **Extensions of Ty2 Meanings**

The relationship between Ty2 meanings and their extensions is axiomatized as follows, where the family of constants $\mathsf{ext}_A : \mathrm{s} \rightarrow (A \rightarrow \mathsf{E}(A))$ is parametrized by the Ty2 meaning types:

a. $\vdash \forall_x \forall_w (\mathsf{ext}_w(x) = x(w))$ (for $x : \mathrm{s} \rightarrow \mathrm{e}$)

b. $\vdash \forall_p \forall_w (\mathsf{ext}_w(p) = p(w))$ (for $p : \mathrm{s} \rightarrow \mathrm{t}$)

c. $\vdash \forall_f \forall_w (\mathsf{ext}_w(f) = \lambda_x \mathsf{ext}_w(f(x)))$ (for $f : A \rightarrow B$, A and B Ty2 meaning types.

Note: we suppress the type parameter, and write ext_w for $\mathsf{ext}(w)$.

(55) **Overall Approach to Interrogative Semantics**

The approach is described in detail in Pollard 2008. Key ideas:

- The analysis of polar questions (after transformation into Ty2) is that of Karttunen 1977: at each world w, an interrogative sentence denotes a set of w-facts (in this case, a singleton).
- For n-ary constituent interrogatives, the denotation at w is a (curried) n-ary function to w-facts. The range of that function is similar to the Karttunen semantics, except that it contains both positive and negative 'true atomic answers.'
- An interrogative meaning of this kind induces an equivalence relation on worlds which is a **refinement** of the Groenendijk-Stokhof (1984) partition semantics.

(56) **Types for Polar Questions**

a. RC meaning type: κ

b. Meaning type of Ty2 transform: $(s \to t) \to (s \to t)$ (property of propositions)

c. Type of Ty2 denotation: $(s \to t) \to t$ (characteristic function of) a (singleton) set of propositions)

d. Example: at w, *Does Chris walk* (or *whether Chris walks*) denotes the singleton set whose member is whichever is true at w, the proposition that Chris walks or the proposition that s/he doesn't.

(57) **Types for Unary Constituent Questions**

a. RC meaning type: κ_1

b. Meaning type of Ty2 transform: $(s \to e) \to (s \to t) \to (s \to t)$ (function from individual concepts to properties of propositions).

c. Type of Ty2 denotation: $(s \to e) \to (s \to t) \to t$ (function from individual concepts to sets of propositions). Technically, the curried version of the characteristic function of a certain binary relation between individual concepts and propositions.

d. Example: at w, *who walks* denotes the (functional) binary relation between individual concepts x and propositions p that obtains just in case x is a w-person and and p is whichever proposition is a w-fact, that x walks or that x does not walk.

(58) **Types for Binary Constituent Questions**

a. RC meaning type: κ_2

b. Meaning type of Ty2 transform: $(s \to e) \to (s \to e) \to (s \to t) \to (s \to t)$ (curried function from pairs of individual concepts to properties of propositions).

c. Type of Ty2 denotation: $(s \to e) \to (s \to e) \to (s \to t) \to t$ (curried function from pairs of individual concepts to sets of propositions). Technically, the curried version of the characteristic function of a certain ternary relation between individual concepts, individual concepts, and propositions.

d. Example: at w, *who likes what* denotes the (functional) ternary relation between individual concepts x and y and propositions p that obtains just in case x is a w-person, y is a w-thing, and p

is whichever proposition is a w-fact, that x likes y or that x does not like y.

(59) **Multiple *Wh*-In Situ vs. Multiple Quantifier Raising**

a. The fact that not all questions have the same type introduces a complexity that does not arise with QR.

b. Since the scope result type of a quantifier is the same as its scope type, we can scope multiple quantifiers one after the other (51).

c. But (for example,) scoping one in-situ *wh*-operator at a proposition produces a unary consituent question, so its type must be $\iota_{\pi}^{\kappa_1}$.

d. So if we want to scope another in-situ *wh*-operator over that unary constituent question to form a binary consituent question, then *its* type must be $\iota_{\kappa_1}^{\kappa_2}$, etc.

d. We will return to this point presently.

(60) **Ty2 Meaning Postulates for Some Standard Logical Constants**

a. $\vdash \mathsf{id}_n = \lambda_Z Z\ (Z : \tau(\kappa_n))$

b. $\vdash \mathsf{and'} = \lambda_p \lambda_q \lambda_w (p(w) \wedge q(w))$

c. $\vdash \mathsf{or'} = \lambda_p \lambda_q \lambda_w (p(w) \vee q(w))$

d. $\vdash \mathsf{not'} = \lambda_p \lambda_w \neg p(w)$

e. $\vdash \mathsf{equals'}_A = \lambda_x \lambda_y \lambda_w (x = y)$

(61) **Ty2 MPs for Some Interrogative Logical Constants**

a. $\vdash \mathsf{whether'} = \lambda_q \lambda_p (p\ \mathsf{and'}\ ((p\ \mathsf{equals'}\ q)\ \mathsf{or'}\ (p\ \mathsf{equals'}\ \mathsf{not'}(q))))$

b. $\vdash \mathsf{which}^0 = \lambda_Q \lambda_P \lambda_x \lambda_p (Q(x)\ \mathsf{and'}\ \mathsf{whether'}(P(x))(p))$

c. $\vdash \mathsf{which}^n = \lambda_Q \lambda_Z \lambda_{x_0} \ldots \lambda_{x_n} \lambda_p (Q(x)\ \mathsf{and'}\ Z(x_0) \ldots (x_n)(p))\ (n > 0)$

d. $\vdash \mathsf{who}^n = \mathsf{which}^n(\mathsf{person'})$

e. $\vdash \mathsf{what}^n = \mathsf{which}^n(\mathsf{thing'})$

8 Chinese Interrogatives

(62) **Types, Categories, and Schemata for Chinese Fragment**

The same as for English (for this fragment anyway).
This improves on the analysis of Pollard 2007a,b which required construction-specific rules for different kinds of in-situ operators.

(63) **Lexicon for Chinese Fragment**

$\vdash$ Zhangsan, Zhangsan' : NP, ι $\dashv$

$\vdash$ xihuan, like' : (NP $\multimap_{\mathrm{C}}$ (NP $\multimap_{\mathrm{S}}$ S), $\iota \rightarrow (\iota \rightarrow \pi)$ $\dashv$

$\vdash$ xi-bu-xihuan, like?' : (NP $\multimap_{\mathrm{C}}$ (NP $\multimap_{\mathrm{S}}$ S), $\iota \rightarrow (\iota \rightarrow \kappa)$ $\dashv$

$\vdash$ xiang-zhidao, wonder'$_n$: S $\multimap_{\mathrm{C}}$ (NP $\multimap_{\mathrm{S}}$ S), $\kappa_n \rightarrow (\iota \rightarrow \pi)$ $\dashv$

$\vdash$ shei, who^0 : NP, $\iota_{\pi}^{\kappa_1}$ $\dashv$

$\vdash$ shei, who^n : NP, $\iota_{\kappa_n}^{\kappa_{n+1}}$ $\dashv$ (for $n > 0$)

$\vdash$ shenme, what^0 : NP, $\iota_{\pi}^{\kappa_1}$ $\dashv$

$\vdash$ shenme, what^n : NP, $\iota_{\kappa_n}^{\kappa_{n+1}}$ $\dashv$ (for $n > 0$)

(64) **Meaning Postulate for an Interrogative Verb Meaning**

$\vdash$ like?' $= \lambda_y \lambda_x$whether'(like'$(y)(x)$)

(65) **Comments on the Chinese Lexicon**

- *xibuxihuan* 'like?' is a partial-reduplicative interrogative verb form, used for forming (both root and embedded) polar questions.
- The meaning of *xiang-zhidao* 'wonder' has to be type-schematized according to the type of question expressed by the sentential complement.
- The meanings of the *sh*-interrogative words have to be type-schematized according to their scope type (and corresponding result type).

(66) **A Simple Chinese Sentence**

a. Zhangsan xihuan Lisi.

b. Zhangsan like Lisi

c. 'Zhangsan likes Lisi.'

d. $\vdash$ ($^{\mathrm{S}}$ Zhangsan (xihuan Lisi $^{\mathrm{C}}$)) : S

e. Ty2: $\vdash$ like'(Lisi')(Zhangsan') : $\tau(\pi)$

(67) **A Chinese Polar Question**

a. Zhangsan xi-bu-xihuan Lisi?

b. Zhangsan like? Lisi

c. 'Does Zhangsan like Lisi?'

d. $\vdash$ ($^{\mathrm{S}}$ Zhangsan (xi-bu-xihuan Lisi $^{\mathrm{C}}$)) : S

e. Ty2: $\vdash$ whether'(like'(Lisi')(Zhangsan')) : $\tau(\kappa_0)$

(68) **A Chinese Unary Constituent Question**

a. Zhangsan xihuan shenme?

b. Zhangsan like who

c. 'What does Zhangsan like?'

d. $\vdash$ (^{S} Zhangsan (xihuan shenme ^{C})) : S

e. RC: $\vdash$ ($\mathsf{what}^0_y((\mathsf{like'}\ y)\ (\mathsf{Zhangsan'}))$: κ_1 $\dashv$

(69) **A Chinese Binary Constituent Question**

a. Shei xihuan shenme?

b. who like what

c. 'Who likes what?'

d. $\vdash$ (^{S} Shei (xihuan shenme ^{C})) : S

e. RC: $\vdash$ ($\mathsf{who}^1_x(\mathsf{what}^0_y((\mathsf{like'}\ y)\ (x)))$: κ_2 $\dashv$

f. RC: $\vdash$ ($\mathsf{what}^1_y(\mathsf{who}^0_x((\mathsf{like'}\ y)\ (x)))$: κ_2 $\dashv$

The ambiguity is inessential: the two functions are the same modulo permutation of their arguments.

(70) **Baker-Type Ambiguity in English**

a. A: Who knows where we bought what?

b. B: Chris does. (Appropriate when *what* scopes to the embeded question.)

c. B: Chris knows where we bought the books, and Kim knows where we bought the records. (Appropriate when *what* scopes to the root question.)

d. The 'overtly moved' *wh*-expressions must scope at their 'surface' positions: *who* can only scope to the root question, and *where* can only scope to the embedded question.

e. But the in-situ *wh*-expression *what* can scope high or low.

(71) **A Chinese Baker-Type Wh-Scope Ambiguity**

a. Zhangsan xiang-zhidao shei xihuan shenme./?

b. Zhangsan wonder who like what

c. $\vdash$ (^{S} Zhangsan (xiang-zhidao (^{S} shei (xihuan shenme ^{C}) ^{C}))) : S

d. $\vdash$ (($\mathsf{wonder'}_2$ ($\mathsf{who}^1_x(\mathsf{what}^0_y((\mathsf{like'}\ y)\ x))))$ Zhangsan') : π $\dashv$
'Zhangsan wonders who likes what.'

e. $\vdash (\mathsf{who}^0_x((\mathsf{wonder'}_1\ (\mathsf{what}^0_y((\mathsf{like'}\ y)\ x)))\ \mathsf{Zhangsan'}) : \kappa_1 \dashv$
'Who does Zhangsan wonder what (that person) likes?'

f. $\vdash (\mathsf{what}^0_y((\mathsf{wonder'}_1\ (\mathsf{who}^0_x((\mathsf{like'}\ y)\ x)))\ \mathsf{Zhangsan'}) : \kappa_1 \dashv$
'What does Zhangsan wonder who likes?'

(72) **The Gist of the Preceding**

a. Both *sh*-expressions are in situ, so they can each scope high or low.

b. If both scope low (71d), then the root sentence expresses a proposition and the embedded sentence expresses a binary question.

c. If one scopes high and the other low (71e,71f), then the root sentence and the embedded sentence both express unary questions.

d. But they cannot *both* scope high, since then the complement sentence would express a proposition, while the first argument of wonder' must be a question.

9 Extending CVG for "Overt Movement"

(73) **The Pretheoretical Term 'Overt Movement'**

- Here, by **overt movement**, we mean the class of phenomena that have been discussed over the past half-century under such rubrics as **extraction** and **Ā-movement**.
- Thus it refers to phenomena such as those discussed by Chomsky (1977) 'On wh-movement' or by Levine and Hukari 2006 (*The Unity of Unbounded Dependency Constructions*).
- So phenomena that have been discussed under the rubrics of **head movement** (e.g. verb inflection, subject-auxiliary inversion) and NP**-movement**/A**-movement** (e.g. passive, raising to subject) are excluded.
- In English, there are both finite and infinitive overt movement constructions, but for expository simplicity, here we consider only the finite constructions.

(74) **Some Examples of Overt Movement**

a. JOHN$_i$, Fido bit t$_i$. [Topicalization]

b. I wonder [who$_i$ Fido bit t$_i$]. [Indirect Question]

c. Who$_i$ did Fido bite t$_i$? [Direct Question]

d. The neighbor [who$_i$ Fido bit t$_i$] was John. [Relative Clause]

e. Felix bit [who(ever)$_i$ Fido bit t$_i$]. [Free Relative]

f. It was John [who$_i$ Fido bit t$_i$]. [Cleft]

g. [Who$_i$ Fido bit t$_i$] was John. [Plain Pseudocleft]

h. [Who$_i$ Fido bit t$_i$] was he bit John. [Amalgamated Pseudocleft]

i. [[The more cats]$_i$ Fido bit t_i], [[the more dogs]$_j$ Felix scratched t$_j$]. [Left and right sides of Correlative Comparatives]

In all these examples, the expression on the left periphery that is coindexed with the trace is called the **filler**, or **extractee**, or **dislocated expression**.

(75) **A Preview of the CVG Theory of Overt Movement**

- We treat traces as variables/hypotheses in the syntactic logic, exactly as semantic variables are treated in the semantic logic.
- Unlike Chomsky's notion of syntactic variables, there is no sense in which they arise from movement (or copying).
- Our schema for overt movement phenomena is a generalized ND formulation of Gazdar's (1979) linking schemata
- HPSG generalized Gazdars's / to the list-valued SLASH feature, and HPSG's Filler-Head Schema also schematized over Gazdar's linking schemata.
- But HPSG's feature-structure encoding obscured the inherent logic behind Gazdar's schemata.
- Whereas in CVG, the counterpart of the category under Gazdar's / (or in HPSG's SLASH feature value) is just the familiar **variable context** of labelled sequent-style ND.

(76) **The ↑ Connective of CG**

- An intended CG counterpart of Gazdar's /, namely ↑, was introduced by Bach (1981).
- This was provided with a Gentzen-sequent-style Right rule by Moortgat (1988), and with an ND-style Introduction rule by Carpenter (1997).
- But the ↑ misses Gazdar's key point: an instance of a linking schema **both** binds the free semantic variable originating from the trace **and** applies to the resulting abstract the meaning of the 'filler' (topicalized prase or -phrase).

(77) **New RC Semantic Schema H for Hypotheses**

$x : A \vdash x : A \dashv$ (x fresh)

This is the same as the usual TLC schema for positing hypotheses. The variable adds itself to the context, and the co-context is empty.

(78) **New RC Semantic Schema O for 'Overt Movement'**

If $\Gamma \vdash a : A^C_B \dashv \Delta$ and $x : A, \Gamma' \vdash b : B \dashv \Delta'$
then $\Gamma; \Gamma' \vdash (a_x b) : C \dashv \Delta, \Delta'$

- O is also mnemonic for 'Outside Assistance'.
- This replaces the Hypothetical Proof schema of TLC, because a hypothesis can be withdrawn only if an operator shows up (the 'outside assistance') to eliminate the implication that **would have** resulted had Hypothetical Proof been available.
- Note that the binding term constructor is the same one used in Responsibility (Cooper Retrieval), but the binding comes from without, not from within.

(79) **Schema O and Gazdar's Linking Schemata**

Note the close similarity to the semantic side of (for example) Gazdar's linking schema for topicalization

$\langle 44, [_{\mathrm{S}}\ \alpha\ \mathrm{S}/\alpha], \lambda_h((\mathrm{S}/\alpha)'](\alpha')\rangle$

where the free variable h originating from the trace is bound, and then the resulting abstract immediately combined with the topic meaning.

(80) **New Kinds of CVG Syntactic Terms**

- There is a denumerable infinity of **traces** (**syntactic variables**) of each of each category.
- There are **binding** syntactic terms of the form $[a_t b]$ where a and b are syntactic terms and t is a trace.
- But there is no λ!

(81) **New CVG Syntactic Schema T for Traces**

$t : A \vdash t : A$ (t fresh)

- This is just the syntactic counterpart of semantic schema H: traces are (literally, not metaphorically) syntactic variables.

- Here CVG differs from Gazdar, who says of his own trace schema $\langle 43, [_{\alpha/\alpha}\ t], h\rangle$
  "And t is a trace. Its only role is phonological ... it serves no semantic function (h is the variable, not t), and for other dialects or languages we could replace t with the empty string e ... or with a proform."
- Gazdar can be forgiven for not having been aware that Mints had extended Curry-Howard to linear logic (in the guise of closed categories) in 1977 (not translated from the Russian till 1981). The inventions of the linear and bilinear lambda calculi (van Benthem 1983 and Buszkowski 1987 respectively) still lay a few years ahead.

(82) **New CVG Syntactic Schema G for 'Overt Movement'**
If $\Gamma \vdash a :^{C}_{B} A$ and $t : A, \Gamma' \vdash b : B$, then $\Gamma; \Gamma' \vdash [a_t b] : C$

- 'G' is mnemonic for 'Gazdar'.
- Here we use a new ternary syntactic category constructor G: if A, B, and C are categories, then $\mathrm{G}[A, B, C]$, usually abbreviated to ${}^{C}_{B}A$, is an **operator** category with **binding** category A, **scope** category B, and **result** category C.
- This generalizes, in ND format, all of Gazdar's linking schemata.
- HPSG's Filler-Head schema also schematized over Gazdar's linking schemata, but the feature-structure encoding obscured the logic underlying them.

(83) **Gazdar vs. the Categorial ↑ Constructor**

- Now look again at the *syntactic* side of Gazdar's topicalization schema
  $\langle 44, [_{\mathrm{S}}\ \alpha\ \mathrm{S}/\alpha], \lambda_h((\mathrm{S}/\alpha)'](\alpha')\rangle$
  The right way to understand Gazdar's / logically is not as an implication but rather as the ND turnstile.
- For example, a verb that could take an S-complement could also take an S/NP-complement.
- Hence the CG ↑ misses the point of Gazdar's linking schemata: they neither introduce nor eliminate an implication.

(84) **Gazdar and CVG's G Constructor**

- Rather than introducing or eliminating an implication, Gazdar's topicalization schema "fills an α gap in an S with an α, resulting in an S."
- In this schema, the "filler" has the same category as the gap, but in Gazdar's schemata for *wh*-relative and interrogative clauses, the filler (but crucially not the gap) bears a WH-feature.
- This reflects that the filler has neither moved from the gap site (*pace* EST/GB) nor copied the gap (*pace* Chomsky 1993).
- So logically, these schemata are really ternary constructors (the type of the gap, the type of the filler's sister, and the type of the mother).
- That is what the G-constructor is intended to capture.
- This is reminiscent of Moortgat's q, but the filler is not in situ.

(85) **Previous CVG Interface Schemata, with Contexts Added**

Schema L (Lexicon)
$\vdash w, c : A, B \dashv$ (for certain pairs $\langle w, c\rangle$ where w is a word of category A and c is a basic constant of type B)

Schema $\mathbf{M_s}$ (Subject Modus Ponens)
If $\Gamma \vdash a, c : A, C \dashv \Delta$ and $\Gamma' \vdash f, v : A \multimap_{\mathrm{S}} B, C \to D \dashv \Delta'$
then $\Gamma; \Gamma' \vdash ({}^{\mathrm{S}}\ a\ f), (v\ c) : B, D \dashv \Delta; \Delta'$

Schema $\mathbf{M_c}$ (Complement Modus Ponens)
If $\Gamma \vdash f, v : A \multimap_{\mathrm{C}} B, C \to D \dashv \Delta$ and $\Gamma' \vdash a, c : A, C \dashv \Delta'$
then $\Gamma; \Gamma' \vdash (f\ a\ {}^{\mathrm{C}}), (v\ c) : B, D \dashv \Delta; \Delta'$

Schema C (Commitment)
If $\Gamma \vdash a, b : A, B^D_C \dashv \Delta$, then $\Gamma \vdash a, x : A, B \dashv b_x : B^D_c; \Delta$ (x fresh)

Schema R (Responsibility)
If $\Gamma \vdash e, c : E, C \dashv b_x : B^D_C; \Delta$ then $\Gamma \vdash e, (b_x c) : E, D \dashv \Delta$
(x free in c but not in Δ)

(86) **New CVG Interface Schemata for Overt Movement**

Schema T (Trace)
$t, x : A, B \vdash t, x : A, B \dashv$ (t and x fresh)

Schema G (Generalized Gazdar Schema)
If $\Gamma \vdash a, d : {}^{C}_{B}\, A, D^{F}_{E} \dashv \Delta$ and $t, x : B, D; \Gamma' \vdash b, e : B, E \dashv \Delta'$
then $\Gamma; \Gamma' \vdash [a_t b], (d_x e) : C, F \dashv \Delta, \Delta'$
(t free in b, x free in e)

N.B.: Contexts are **lists** of **pairs** of a trace and a semantic variable. This captures the **Prohibition on Crossed Dependencies**.

10 Topicalization in English

(87) **A New Grammatical Function for Phrasal Affixation**

- We add to the inventory of grammatical function names the name AFFIX (abbr. A), mnemonic for '(phrasal) affixation'.
- As with other grammatical functions, we add a new flavor of Modus Ponens to the syntactic (and interface) schemata (the Elimination rule for $\multimap_{\text{A}}$).
- This schema will be used to analyze (e.g) intonationally realized information-structural affixes, Japanese and Korean case markers, Chinese sentence particles, English possessive *-'s*, etc.

(88) **Syntactic Analysis of Topicalization in CVG**

- We add to the lexicon a topicalization phrasal-affix schema
  $\vdash \mathsf{top}, \mathsf{top'} : A \multimap_{\text{A}} {}^{\text{T}}_{\text{S}}A, B \rightarrow B^{\pi}_{\pi} \dashv$
- The syntax-phonology interface will specify that top is optionally realized as a contrastive-topic pitch accent on the realization of the A.
- top changes the phrase it affixes to into a syntactic operator: it is CVG's analog of the notion of a 'movement trigger' in TG.
- We remain agnostic for now whether the category T of topicalized sentences is distinct from S.

(89) **Semantic Analysis of Topicalization in CVG**

- The assumption that the meaning of a topicalized sentence is just a proposition is a simplification in the absence of a concrete type-theoretic embodiment of information-structural concepts such as contrastive topic.

- Under the RC-toTy2 transform, the semantic operator $\mathsf{top'}$ is mapped to a family (parametrized by Ty2 meaning types B) of constants of type $B \to (B \to (\mathrm{s} \to \mathrm{t})) \to (\mathrm{s} \to \mathrm{t})$.
- Intuitively, $\mathsf{top'}(b)(P)$ means something like '$P(b)$, but for certain other B's, x, contextually associated with b, $\mathsf{whether'}(P(x))$ remains unresolved.'
- This approach (I think) gets at the intuition that 'structured propositions' try to get at, but as far as its type goes, $\mathsf{top'}(b)(P)$ is just a proposition.

(90) **CVG Analysis of *Sandy, Kim likes***

$\vdash [(\mathsf{Sandy\ top}\ ^{\mathrm{A}})_t(^{\mathrm{S}}\ \mathsf{Kim\ (likes}\ t\ ^{\mathrm{C}}))],$
$((\mathsf{top'\ Sandy'})_x((\mathsf{like'}\ x)\ \mathsf{Kim'})) : \mathrm{T}, \pi \dashv$

11 English Interrogatives

(91) **Lexicon for English Fragment**

$\vdash \mathsf{Kim}, \mathsf{Kim'} : \mathrm{NP}, \iota \dashv$

$\vdash \mathsf{likes}, \mathsf{like'} : (\mathrm{NP} \multimap_{\mathrm{C}} (\mathrm{NP} \multimap_{\mathrm{S}} \mathrm{S}), \iota \to (\iota \to \pi) \dashv$

$\vdash \mathsf{whether}, \mathsf{whether'} : (\mathrm{S} \multimap_{\mathrm{C}} \mathrm{S}, \pi \to \kappa) \dashv$

$\vdash \mathsf{wonders}, \mathsf{wonder'}_n : \mathrm{S} \multimap_{\mathrm{C}} (\mathrm{NP} \multimap_{\mathrm{S}} \mathrm{S}), \kappa_n \to (\iota \to \pi) \dashv$

$\vdash \mathsf{who}_{\mathsf{filler}}, \mathsf{who}^0 :^{\mathrm{S}}_{\mathrm{S}}\ \mathrm{NP}, \iota^{\kappa_1}_{\pi} \dashv$

$\vdash \mathsf{who}_{\mathsf{in\text{-}situ}}, \mathsf{who}^n : \mathrm{NP}, \iota^{\kappa_{n+1}}_{\kappa_n} \dashv (\text{for } n > 0)$

$\vdash \mathsf{what}_{\mathsf{filler}}, \mathsf{what}^0 :^{\mathrm{S}}_{\mathrm{S}} \mathrm{NP}, \iota^{\kappa_1}_{\pi} \dashv$

$\vdash \mathsf{what}_{\mathsf{in\text{-}situ}}, \mathsf{what}^n : \mathrm{NP}, \iota^{\kappa_{n+1}}_{\kappa_n} \dashv (\text{for } n > 0)$

(92) **Comments on the English Lexicon**

- Unlike Chinese, (embedded) polar interrogatives are formed by 'complementizing' declarative sentences with *whether*, which has the same meaning as the Chinese interrogative reduplicative verbal affix.
- The verb *wonder* is type-schematized according to the type of question expressed by the sentential complement.
- The meanings of the *wh*-interrogative words are type-schematized as in Chinese, with one crucial difference.

(93) **The Difference between Chinese and English**

- The main difference between English and Chinese, at least as far as interrogatives are concerned, is a lexical one: in English, but not in Chinese, the interrogative pronouns are syntactically ambiguous.
- For example, there is only one Chinese syntactic word *shei* 'who', namely $\mathsf{shei} : \mathrm{NP}$.
- But in English, the *who* with meaning $\mathsf{who}^0 : \iota^{\kappa}_{\pi}$, the one that 'gets the ball rolling', is **not** an NP but rather the syntactic operator $\mathsf{who}_{\mathsf{filler}} :^{\mathrm{S}}_{\mathrm{S}} \mathrm{NP})$.
- This is the wrong category to be a subject or a complement, but the right category to be a filler in Schema G.
- On the other hand, the English *who* that expresses all the all the other meanings ($\mathsf{who}_n : \iota^{\kappa_{n+1}}_{\kappa_n}$, for $n > 0$), namely $\mathsf{who}_{\mathsf{in\text{-}situ}}$, is an NP just like Chinese shei, so it is always in situ.

(94) **A Simple English Sentence**

a. Kim likes Sandy.

b. $\vdash (^{\mathrm{S}}\ \mathsf{Kim}\ (\mathsf{likes}\ \mathsf{Sandy}\ ^{\mathrm{C}})) : \mathrm{S}$

c. Ty2: $\vdash \mathsf{like'}(\mathsf{Sandy'})(\mathsf{Kim'}) : \tau(\pi)$

(95) **An English Embedded Polar Question**

a. whether Kim likes Sandy

b. $\vdash (\mathsf{whether'}\ (^{\mathrm{S}}\ \mathsf{Kim}\ (\mathsf{likes}\ \mathsf{Sandy}\ ^{\mathrm{C}}))\ ^{\mathrm{C}}) : \mathrm{S}$

c. Ty2: $\vdash \mathsf{whether'}(\mathsf{like'}(\mathsf{Sandy'})(\mathsf{Kim'})) : \tau(\kappa_0)$

(96) **An English Embedded Constituent Question**

a. what Kim likes

b. $\vdash [\mathsf{what}_{\mathsf{filler}\ t}(^{\mathrm{S}}\ \mathsf{Kim}\ (\mathsf{likes}\ t\ ^{\mathrm{C}}))] : \mathrm{S}$

c. RC: $\vdash (\mathsf{what}^0_y((\mathsf{like'}\ y)\ (\mathsf{Kim'})) : \kappa_1 \dashv$

(97) **A English Binary Constituent Question**

a. who likes what?

b. $\vdash [\mathsf{who}_{\mathsf{filler}\ t}(^{\mathrm{S}}\ t\ (\mathsf{likes}\ \mathsf{what}_{\mathsf{in\text{-}situ}}\ ^{\mathrm{C}}))] : \mathrm{S}$

c. RC: $\vdash (\mathsf{what}^1_y(\mathsf{who}^0_x((\mathsf{like'}\ y)\ (x))) : \kappa_2 \dashv$

No ambiguity, not even an inessential one.

(98) **Baker-Type Ambiguity in English (Review)**

a. A: Who wonders who likes what?

b. B: Chris does. (Appropriate when *what* scopes to the embeded question.)

c. B: Chris wonders who likes the books, and Kim wonders who likes the records. (Appropriate when *what* scopes to the root question.)

d. The 'overtly moved' *wh*-expressions must scope at their 'surface' positions: *who* can only scope to the root question, and *where* can only scope to the embedded question.
Now we know why: because syntactic operators (fillers in Schema G) are interpreted 'in place'.

e. But the in-situ *wh*-expression *what* can scope high or low.
Now we know why: in-situ operators are stored, and have multiple options as to where they are retrieved.

(99) **CVG Analysis of Baker-Type Ambiguity in English**

a. Who wonders who likes what?

b. $\vdash [\mathsf{who}_{\mathsf{filler}\ t}({}^{\mathrm{S}}\ t\ (\mathsf{wonders}\ [\mathsf{who}_{\mathsf{filler}\ t'}({}^{\mathrm{S}}\ t'\ (\mathsf{likes}\ \mathsf{what}_{\mathsf{in\text{-}situ}}\ {}^{\mathrm{C}}))]\ {}^{\mathrm{C}}))] : \mathrm{S}$

c. $\vdash (\mathsf{who}^0_x((\mathsf{wonder'}_2\ (\mathsf{what}^1_y(\mathsf{who}^0_z((\mathsf{like'}\ y)\ z))))\ x)) : \pi \dashv$
(E.g. Chris wonders who likes what.)

d. $\vdash (\mathsf{what}^1_y(\mathsf{who}^0_x((\mathsf{wonder'}_1\ (\mathsf{who}^0_z((\mathsf{like'}\ y)\ z)))\ x)))) : \pi \dashv$
(E.g. Chris wonders who likes the books, and Kim wonders who likes the records.)

(100) **The Gist of the Preceding**

a. Both instances of *who* are fillers, so they must scope 'in place'.

b. But *what* is in situ, so it can scope high or low.

12 Conclusion

(101) **Summary**

- CVG is purely derivational like ACG, weakly syntactocentric like HPSG, and parallel like both.
- It is a synthesis of ideas developed by phrase structure grammarians and categorial grammmarians
- The syntax-semantics interface recursively specifies a set of pairs consisting of a syntactic proof and a semantic proof.
- The syntactic logic is a kind of noncommutative multilinear logic with the usual hypothetical proof rule replaced by a generalized Gazdar-style linking schema.
- The semantic logic (RC) is a kind of (commutative) linear logic with the usual hypothetical proof rule replaced by (1) ND formulations of Cooper storage and retrieval (i.e. by 'q but in the semantics'), and (2) a semantic counterpart of the Gazdar-style linking schema.
- There is a simple, explicit transform from RC to Ty2 (limited to terms with empty Cooper stores).
- Quantifier raising, topicalization, and both 'moved' and 'in situ' constituent questions are all handled smoothly with no construction-specific rules.
- Traces are literally syntactic variables.
- Syntactic binding operators never occupied the trace position, nor are they copies of traces.
- Semantic binding operators (*qua* terms) 'move' only in the sense that a copy of the term can be found in the Cooper store of every node in the proof tree between the storage and the retrieval, but in no sense is the operator a copy of the variable it binds.