Covert Movement in Logical Grammar

Carl Pollard*

INRIA-Lorraine and Ohio State University
pollard@ling.ohio-state.edu

1 Introduction

From the mid-1970s until the emergence of Chomsky's Minimalist Program (MP, Chomsky 1995) in the 1990s, the mainstream of research on natural-language syntax in much of the world embraced a theoretical architecture for syntactic derivations that came to be known as the **T-model**. According to this model, which underlay Chomsky's (1976, 1977) Extended Standard Theory (EST) of the 1970s and its successor, the Government-Binding (GB) Theory [Chomsky 1981) of the 1980s and early 1990s, a tree called a **deep structure** (DS) is generated from lexical entries by essentially context-free **base rules**. The DS is then converted into a **surface structure** (SS) by **transformations**, destructive structural operations that can delete, copy, or (most significantly for us) move subtrees. From SS, the derivation branches (the two arms of the T): in one direction the SS is further transformed into a **phonetic form** (PF), which determines what the expression being analyzed sounds like, and in the other direction the SS is transformed into a **logical form** (LF), which determines what the expression means.

* For helpful discussion and comments on earlier stages of this work, I am grateful to Chris Barker, Patrick Blackburn, Wojciech Buszkowski, Robin Cooper, David Dowty, Jonathan Ginzburg, Philippe de Groote, Jirka Hana, Martin Jansche, Ruth Kempson, Brad Kolb, Yusuke Kubota, Alain Lecomte, Tim Leffel, Jim Lambek, Scott Martin, Vedrana Mihalicek, Glyn Morrill, Michael Moortgat, Reinhard Muskens, Guy Perrier, Andy Plummer, Ken Shan, Elizabeth Smith, Chris Worth, workshop participants at ESSLLI Workshop on New Directions in Type-Theoretic Grammar (Dublin 2007), the Fourth Workshop on Lambda Calculus and Formal Grammar (Nancy, 2007), the Colloque en l'honneur d'Alain Lecomte (Pauillac 2007), the Second Workshop on Types, Logic, and Grammar (Barcelona, 2007), the NaTAL Workshop on Semantics and Inference (Nancy 2008), the ESSLLI Workshop on Continuationd and Symmetric Calculi (Hamburg 2008), and audiences at the Séminaire de l'UMR 7023, CNRS/ Université de Paris 8 (2008), the Séminaire Calligramme, INRIA-Lorraine (2008), and the Centre de Lingüística Teòrica, Universitat Autònoma de Barcelona (2008). In addition, I benefited greatly from the comments of two anonymous referees. For their help in providing the conditions that made this research possible, I am grateful to Carlos Martin Vide, Philippe de Groote, and to the Department of Linguistics and College of Humanities of Ohio State University. The research reported here was supported by grant no. 2006PIV10036 from the Agència de Gestió d'Ajuts Universitaris i de Recerca of the Generalitat de Catalunya.

In the T-model, the tranformations that convert DS to SS are called **overt**, because their effects are (at least potentially) audible (since the branch of the derivation that leads to PF is yet to come). The prototypical case of overt movement is **overt wh-movement** in languages like English, where constituent questions are formed (so the theory goes) by moving a wh-expression (or, in so-called **pied-piping** constructions, an expression properly containing a wh-expression) to the left periphery of a clause. Since both PF and LF are derived from SS, this movement is subsequently reflected in both how the sentence sounds, and what it means:

(1) **Overt Wh-Movement in the T-Model**

a. I wonder who Chris thinks Kim likes.

b. DS: (I wonder (Chris thinks (Kim likes who)))

c. SS: (I wonder (who_t (Chris thinks (Kim likes t))))

d. LF: (I wonder (who_x (Chris thinks (Kim likes x))))

Here, the **wh-operator** *who* occupies an **argument (A)** position at DS. After overt movement, it occupies a **nonargument ($\bar{\text{A}}$)** position in SS on the left periphery of one of the clauses that contained it; in this sentence, the only clause it can move to is the middle one (with subject *Chris*), because the verb *wonder* is the kind of verb that requires an interrogative complement clause. When *who* moves, it leaves behind a **trace** or **syntactic variable** (here, t), which it **binds** at SS; this is essentially the same position it will occupy at LF. Now since derivations branch to PF (and LF) *after* SS, the movement of *who* has an audible reflex (you hear it in the position it moved to). And finally, during the SS-to-LF derivation, a **rule of construal** replaces t with a **logical variable** (here, x), which is bound by *who* at LF.

Now nobody with even a rudimentary knowledge of lambda calculus or predicate logic could fail to notice that the SS in (1c) and the LF in (1d) look a lot like formal terms containing operators that bind variables. But, at least as far as I know, no logician has ever suggested that λ's, or $\exists$'s, or $\forall$'s, actually start out in the position of the variables they bind, and then move to the left. So one might well ask why transformational grammarians, right down to the present day, believe that binding operators in NL do. At least 30 years ago, practicioners of **categorial grammar (CG)** (e.g. David Dowty, Emmon Bach) and **phrase structure grammar (PSG)**(e.g. Gerald Gazdar, Geoff Pullum) started asking this very question, and in the intervening decades researchers in these frameworks have proposed a wealth of carefully thought out theories in which NL binding operators do **not** move. We will come back to this.

By contrast with overt movement (within the T-model), transformations that convert SS to LF are called **covert** because they take place too late in the derivation—after the SS branch point—to have a reflex at PF. One standardly assumed covert movement is **quantifier raising** (QR, May 1977 and 1985), which moves a quantificational NP (QNP) to a position in LF (reflective of its semantic scope) higher than the one it occupied at SS.

(2) **Covert Wh-Movement in the T-Model: QR**

a. I know Chris thinks Kim likes everyone.

b. DS: (I know (Chris thinks (Kim likes everyone)))

c. SS: (I know (Chris thinks (Kim likes everyone))) [no change]

d. LF (narrow scope reading): (I know (Chris thinks (everyone$_x$ (Kim likes x))))

e. LF (medium scope reading): (I know (everyone$_x$ (Chris thinks (Kim likes x))))

f. LF (wide scope reading): (everyone$_x$ (I know (Chris thinks (Kim likes x))))

Here, the QNP *everyone* occupies an **argument (A)** position at DS, and nothing happens to it between DS and SS (no overt movement). But after covert movement, it occupies a **nonargument (Ā)** position in LF on the left periphery of one of the clauses that contained it. Now when *everyone* moves, it leaves behind a **logical variable** (here, x), which it **binds** at LF. But since derivations branch after SS to PF and LF, and the movement of *everyone* is on the the SS-to-LF branch, it has no audible reflex (you hear it in its pre-movement position).

Another standardly assumed covert movement is **covert wh-movement** in languages like Chinese (Huang 1982, Pesetsky 1987)[1]. Covert wh-movement is supposed to be essentially the same as overt wh-movement, except that, since—like QR— it takes place *after* the SS branch point, it is *heard* just as if it had never moved (or, to use the syntactician's term of art, it remains **in situ**).

(3) **Covert Wh-Movement in the T-Model: *Wh*-in-Situ**

a. Zhangsan xiang-zhidao shei mai-le shenme. [Chinese]

b. Zhangsan wonder who bought what [English word-for-word gloss]

c. DS: (Zhangsan xiang-zhidao (shei mai-le shenme))

d. SS: (Zhangsan xiang-zhidao (shei mai-le shenme)) [no change]

e. LF (*shei* and *shenme* both narrow):
(Zhangsan xiang-zhidao (shei$_x$ shenme$_y$ (x mai-le y)))
'Zhangsan wonders who bought what'

f. LF (*shenme* narrow, *shei* wide):
(shei$_x$(Zhangsan xiang-zhidao (shenme$_y$ (x mai-le y))))
'Who does Zhangsan wonder what (s/he) bought?'

g. LF (*shenme* wide, *shei* narrow):
(shenme$_y$(Zhangsan xiang-zhidao (shei$_x$ (x mai-le y))))
'What does Zhangsan wonder who bought?'

[1] But see Aoun and Li 1993 for a dissenting view (that Chinese *wh*-movement is overt movement of an inaudible operator, with the *wh*-expressions as bindees, not binders.)

Here, as with QR, there is no change between DS and SS. Each of the *wh-* (or, in Chinese, *sh-*)operators can scope to any of the clauses containing it. However, in this example, at least one of them must scope to the lower clause, since the clausal complement of the verb *xiang-xhidao* 'wonder' has to be a question.

In fact, even languages like English with overt wh-movement also have in situ wh, in two different respects. First, in **multiple constituent questions**, all but the leftmost wh-expression remain in situ. And second, in cases of pied piping, the wh-expression that is properly contained within the moved constituent remains in situ, relative to the displaced constituent that contains it. In this paper, however, we will limit our attention henceforth to phenomena that transformational grammar (TG) has analyzed purely in terms of covert movements.

In the rest of the paper, we sketch an approach to so-called covert phenomena in which (as in logic) binding operators never move. (For the extension of this approach to so-called overt movement phenomena, see Pollard 2008b.)

2 Toward a New, Nontransformational Synthesis

The T-model has long since been abandoned. Within the Chomskyan syntactic tradition, the Minimalist Programm (MP, Chomsky 1995) provides much more flexibility than EST or GB did, by discarding the notions of DS and SS. Instead, merges (corresponding to EST/GB base rules) need not all take place before any moves do. And the possibility of multiple branch points in a single derivation ('Spell-outs') means that not all overt moves must occur 'lower' in the derivation than any of the covert ones. These are not exactly negative developments; but it is well worth noting that, had transformational grammarians followed the lead of CG and PSG practicioners from the 1970s on in informing their theory by ideas from logic (as opposed to logic metaphors), the architectural problems of EST/GB that the MP has sought to repair could have been addressed much early, or even avoided altogether. Here are a few examples.

First, in EST/GB, as noted above, LF is derived from SS. But an LF looks a lot like a semantic lambda-term, and so, in light of the Curry-Howard (types as formulas, terms as proofs) conception (Curry 1958, Howard 1980), we should be able to think of it as an (intuitionistic) proof in its own right. So there is no reason why it has to be derived from SS (or anything else).

Second, also as noted above, an EST/GB labelled bracketing, which typically contains traces (syntactic variables) and inaudible operators which bind them, also looks a lot like a lambda term. But by then (1970s to early 1980s), Lambek (1958) had already long since proposed that NL syntax be formulated in terms of a substructural proof theory. Moroever the idea of extending the Curry-Howard conception to substructural logics was continually being rediscovered[2]; so, in hindsight at least, it is easy perceive these labelled bracketings as Curry-Howard terms for some resource-sensitive logic or other. But in that case, linguists should think of NL syntactic trees as *proof trees*, as Moortgat

[2] E.g. Mints 1977, van Benthem 1983, Buszkowski 1987, Jay 1989, Benton et al. 1992, Wansing 1992, Gabbay and de Queiroz 1992, Mackie et al. 1993).

(1988) and other categorial grammarians had already realized in the mid-to-late 1980s, not as *structures* whose subtrees can be deleted, copied, or moved by transformations (and whose internal structural configurations could be relevant in the formulation of linguistically significant generalizations).

Third (given the preceding), there is no need to stipulate a Strict Cycle Condition (Chomsky 1976) on rule application (roughly, that once a rule has applied to a given tree, it is already too late for any rule to apply solely to one of that tree's proper subtrees), for the simple reason that a proof cannot go back and change earlier parts of itself!

And fourth, also in hindsight, it is clear that the notion of SS is not only unnecessary but pernicious. That is because SS is the stage of the derivation at which all base rule applications (merges) have taken place but none of the transformational rule applications (moves). In proof theoretic terms, what SS amounts to is a point in a proof subsequent to which only instances of Hypothetical Proof (but not Modus Ponens) are admitted! But there is no requirement on proofs that all instances of Modus Ponens appear lower in the proof tree than all instances of Hypothetical Proof, just as there is no well-formedness condition on lambda terms that all the abstractions occur on the left periphery of the term.

If these observations are on the right track, then the syntax and semantics of NL expressions are both proofs in their own right. But then, a grammar should not be in the business of tranforming syntax into semantics; rather, it should be specifying which syntax-semantics *pairs of proofs*[3] go together. To put it another way, the syntax-semantics interface should be at once **purely derivational** and **parallel**. Here, by *purely* derivational, we mean simply that derivations are *proofs*, as opposed to nondeterministic algorithms that build arboreal structures via successive destructive modification. And by *parallel*, we mean that there are separate proofs theories that provide, respectively, candidate syntactic and semantic proofs; whereas it is the job of the syntax-semantics interface to recursively define the set of proof pairs that belong to the language in question.

The pure derivationality of the proposed approach comes straight out of CG, and the syntactic proof theory we will adopt below will be readily taken for what it is, a variant of (multimodal) applicative categorial grammar. However, the mainstream of CG[4] has eschewed parallelism, in favor of the **functional** approach to semantic interpretation bequeathed by Montague, which mandates that there can never be a purely semantic ambiguity. Rather, on the functional approach, there must be a *function* from syntactic proofs/terms[5] to semantic

[3] Or triples, if phonology is also taken into account.

[4] E.g.van Benthem 1983, Lambek 1988, Morrill 1994, Steedman 1996, Moortgat 1997, Carpenter 1997, Jacobson 1999, de Groote 2001b, Ranta 2002, Muskens 2003, Pollard 2004, Anoun and Lecomte 2007, Bernardi and Moortgat 2007.

[5] Or, in the case of Montague, analysis trees.

proofs/terms; or, to put it another way, all meaning differences must be disambiguated in the syntax.[6]

But I am not aware of any scientific basis for requiring that the relation between syntactic derivations and semantic ones be a function. Indeed, there is a long tradition[7] which rejects the premise that the syntax-semantics relation is a function from the former to the latter. I will refer to this tradition as the **parallel** approach to the syntax-semantics interface. The framework I will be using below, called **Convergent Grammar (CVG)**, while purely derivational, also lies squarely within this parallel tradition.[8]

In fact, the idea of a purely derivational parallel grammar architecture has already been proposed independently and considerably earlier in this decade by Lecomte and Retoré (Lecomte and Retoré 2002, Lecomte 2005), and there are numerous points of similarity between their approach and CVG. However, unlike their approach, which is part of a larger intellectual enterprise (**categorial minimalism**) which seeks to bring about a marriage of CG and MP, the intellectual tradition to which CVG belongs is one that parted company with (to use Culicover and Jackendoff's term) mainstream generative grammar (MGG) more than three decades ago. As will be made clear shortly, CVG is really a proof-theoretic embodiment not of minimalism but rather of the storage and retrieval technology proposed by Cooper (1975, 1983) as an alternative to the then-current EST/GB.

The kinds of ambiguities associated with the so-called covert-movement phenomena, illustrated above in (2) and (3), bear directly on the functional vs. parallel issue. Indeed, on parallel approaches, they readily lend themselves to analyses that locate the ambiguities wholly in the semantics, rather than complicating the syntax for the mere sake of preserving the (putative) functionality of the syntax-semantics interface at any cost. To put it simply, it is entirely permissible, at a certain point in a pair of simultaneous derivations (one syntactic; one semantic), to do something on the semantic side while doing nothing at all on the syntactic side. And as we will see shortly, the Cooper-inspired storage and

[6] There is a trivial respect in which any relational syntax-semantics interface can be rendered functional by allowing *sets* of usual meanings to serve as 'meanings', since there is a canonical correspondence between binary relations and functions from the domain of the relation to the powerset of the codomain (the category of relations is the Kleisli category of the powerset monad on sets). But linguists generally require of meanings, however they are modelled, that they provide a deterministic interpretation for contextualized utterances. Thus, we rule out as meanings nondetermistic (or underspecified) representations (as in the MRS (minimal recursion semantics) employed in some versions of head-driven phrase structure grammar (HPSG)) that have to be postprocessed to resolve scopal ambiguities.

[7] See, e.g. Cooper 1975 and 1983, Bach and Partee 1980, Hendriks 1993, Pollard and Sag 1994, Lecomte and Retoré 2002, Lecomte 2005, and Culicover and Jackendoff 2005.

[8] This is scarcely surprising, since it originated as an effort to reformulate HPSG along type-theoretic lines (Pollard 2004).

retrieval rules in terms of which we analyze covert movement are of this precise character.

3 Syntax, Semantics, and their Interface

For present purposes, we take a CVG to consist of three things: (1) a **syntax**, (2) a **semantics**, and (3) a **syntax-semantics interface** (hereafter, **interface** *simpliciter*).[9] For the fragment developed here, we can take the syntax to be a proof theory for a simple multi-applicative categorial grammar.[10] The semantics will be another proof theory closely related to the familiar typed lambda calculus (TLC). And the interface will recursively define a set of pairs of proofs. The two proof theories are both presented in the Gentzen-sequent style of natural deduction with Curry-Howard proof terms (see e.g. Mitchell and Scott 1989), because this style of proof theory is visually easy to relate to EST/GB-style or HPSG-style linguistic analyses.

3.1 Semantics

Rather than the familiar TLC, we employ a new semantic calculus RC (the calculus of **Responsibility and Commitment**[11] which, we argue, is better adapted to expressing the semantic compositionality of natural language. (But we will also provide a simple algorithm for transforming RC semantic terms into TLC, more specifically, into Ty2.) Here we present only the fragment of RC needed to analyze covert movement; the full calculus, with the two additional schemata needed to analyze overt movement, is presented in Pollard 2008b.

Like TLC, RC has types, terms, and typing judgments. One important difference is that in TLC, the **variable context** of a typing judgment is just a set of variable/type pairs, written to the left of the turnstile. But an RC typing judgment has a **Cooper store**, written to the right and demarcated by a **co-turnstile** $\dashv$:

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

$$\vdash a : A \dashv \Delta$$

The Cooper store is also called the **variable co-context**[12]; the 'co-' here is mnemonic not only for 'Cooper'; but also for 'Commitment' (for reasons to

[9] For the simple fragment developed here, it is easy to read the word order off of the syntactic analyses (proof terms). But to do serious linguistics, we also will require a **phonology** and a **syntax-phonology interface**. Thus CVG is **syntactocentric**, in the sense that syntax has interfaces to phonology and semantics, but only **weakly** so, in the sense that the relations defined by the two interfaces need not be functions.

[10] But in order to extend the theory to cover so-called overt movement phenomena, we will need to add some form of hypothetical reasoning to the syntactic proof theory (Pollard 2008b).

[11] See Pollard 2008a and references cited there for background and discussion.

[12] The full RC calculus, including the schemata for analyzing overt movement, also employs ordinary variable contexts to the left of the turnstile.

be explained presently), for 'Covert Movement', and for 'Continuation' (since the operators stored in them will scope over their own continuations). Thus a judgment like (4) is read 'the (semantic) term a is assigned the (semantic) type A in the co-context Δ.'

(5) **RC Semantic Types**

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

b. If A and B are types, then $A \to B$ 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.[13]

(6) **Basic Semantic Types**

For present purposes, we use three basic semantic types:
ι (individual concepts), π (propositions), and κ (polar questions).[14]

(7) **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. $A_\epsilon =_{\mathrm{def}} A$

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

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

(8) **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 $\mathrm{q}[\mathrm{NP}, \mathrm{S}, \mathrm{S}]$ and semantic type $(\iota \to \pi) \to \pi$, for us it has category (simply) NP and semantic type ι^π_π.[15].

(9) **RC Semantic Terms**

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

---

[13] That is, a term a of type $\mathrm{O}[A, B, C]$ binds a variable x of type A in a term of type B, resulting in a term $a_x b$ of type C.

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

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

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

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

d. There are **binding** semantic 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!

(10) **Cooper Stores**

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.

Then the rule schemata of RC are the following:

(11) **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 (see (26).

(12) **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 natural-deduction (ND) Modus Ponens, except that co-contexts have to be propagated from premisses to conclusions.

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

(13) **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.

b. It generalizes Carpenter's (1997) Introduction rule for Moortgat's (1988) $\Uparrow$ (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**.

(14) **Semantic Schema R (Responsibility)**

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

a. This is a straightforward ND formulation of Cooper retrieval.

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.

To give the reader a familiar point of reference, we provide a transform of RC into the standard higher-order semantic representation language Ty2 (Gallin 1975).[16] We follow Carpenter (1997, section 11.2) in using individual concepts as the basic type for NPs. But we use use the Gallin/Montague names for the basic types e (entities, Carpenter's individuals), t (truth values, Carpenter's booleans), and s (worlds), rather than Carpenter's Ind, Bool, and World respectively. Hence our (and Montague's) type $\mathrm{s} \to \mathrm{e}$ for individual concepts corresponds to Carpenter's type $\mathrm{World} \to \mathrm{Ind}$.[17]

We also follow Carpenter's convention that functional meaning types take their world argument last rather than first, e.g. the type for an intransitive verb is $(\mathrm{s} \to \mathrm{e}) \to \mathrm{s} \to \mathrm{t}$ (the transform of RC type $\iota \to \pi$) rather than $\mathrm{s} \to (\mathrm{s} \to \mathrm{e}) \to \mathrm{t}$, so that the verb meaning combines with the subject meaning by ordinary function application.

The price, well worth paying, is that, except for individual concepts and propositions, our Ty2 meanings are technically not intensions (functions from worlds). Consequently the extension at a world w of a Ty2 meaning is defined by recursion on types as follows:

(15) **Ty2 Meaning Types**

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

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

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

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

These are defined as follows:

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

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

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

(17) **Extensions of Ty2 Meanings**

The relationship between Ty2 meanings and their extensions is axiomatized as follows, where the family of constants $\mathsf{ext}_A : \mathrm{s} \to A \to \mathsf{E}(A)$ is parametrized by the Ty2 meaning types:[18]

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

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

[16] This transform is not a proper part of our framework, but is provided in order to show that familiar meaning representations can be algorithmically recovered from the ones we employ. Readers who are not concerned with this issue can just ignore this transform.

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

[18] We omit the type subscript A on ext_A when it is inferrable from context. Moreover we abbreviate $\mathsf{ext}(w)$ as ext_w.

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

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

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

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

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

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

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

(19) **The Transform τ on Terms**

a. Variables and basic constants are unchanged except for their types. (We make abundant use of meaning postulates, e.g. (20) rather than giving basic constants nonbasic transforms.)

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. $\tau((a_x b)) = \tau(a)(\lambda_x \tau(b))$

(20) **Ty2 Meaning Postulates for Generalized Quantifiers**

$\vdash \mathsf{every'} = \lambda_Q \lambda_P \lambda_w \forall_x (Q(x)(w) \to P(x)(w))$

$\vdash \mathsf{some'} = \lambda_Q \lambda_P \lambda_w \exists_x (Q(x)(w) \wedge P(x)(w))$

$\vdash \mathsf{everyone'} = \mathsf{every'}(\mathsf{person'})$

$\vdash \mathsf{someone'} = \mathsf{some'}(\mathsf{person'})$

3.2 Syntax

For the fragment developed here, our syntactic calculus is just a simple multimodal applicative CG.[19] Again, there are types, now called **(syntactic) categories**, terms, and typing judgments, which have the form

(21) **Format for CVG Syntactic Typing Judgments**
$\vdash a : A$
read 'the (syntactic) term a is assigned the category A.'

(22) **CVG Categories**

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

[19] But to analyze overt movement, it will have to be extended with schemata for traces and syntactic binding by 'overtly moved' syntactic operators (Pollard 2008b).

b. If A and B are categories, so are $A \multimap_{\mathrm{F}} B$, where F belongs to a set F of **grammatical function names**[20]; these are called **functional** categories with **argument** category A and **result** category B.

(23) **Basic Categories**
For now, just S and NP.

(24) **Functional Categories**
We start off with the grammatical function names S (subject) and C (complement).[21] Others will be added as needed.

(25) **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 $(^{\mathrm{F}}\ f\ a)$

(26) **(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.

(27) **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.

The CVG syntactic rule schemata are as follows:

(28) **Syntactic Schema W (Words)**
$\vdash w : A$ (w a syntactic word of category A)

(29) **Syntactic Schema $\mathbf{M_S}$ (Subject Modus Ponens)**
If $\vdash a : A$ and $\vdash f : A \multimap_{\mathrm{S}} B$, then $\vdash (^{\mathrm{S}}\ a\ f) : B$

(30) **Syntactic Schema $\mathbf{M_C}$ (Complement Modus Ponens)**
If $\vdash f : A \multimap_{\mathrm{C}} B$ and $\vdash a : A$, then $\vdash (f\ a\ ^{\mathrm{C}}) : B$

[20] Thus grammatical functions are abstract tectogrammatical primitives, and not defined in terms of word order, phonology, or the positions in which they occur in proof trees. And so the role grammatical functions play in CVG is strongly analogous to the role that they play in such frameworks as HPSG, lexical-functional grammar (LFG), and relational grammar (RG). Multiple modes of implication can be replaced by a single linear implication (see de Groote et al. 2009 for details), at the expense of considerably elaborating the set of basic types.

[21] Here CVG betrays its HPSG pedigree.

3.3 The CVG Syntax-Semantics Interface

The interface recursively specifies which syntactic proofs are paired with which semantics ones. Unsurprisingly, the recursion is grounded in the lexicon:

(31) **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)

The following two schemata are essentially ND reformulations of HPSG's Subject-Head and Head-Complement schemata:

(32) **Interface Schema $\mathbf{M_S}$ (Subject Modus Ponens)**

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

(33) **Interface Schema $\mathbf{M_C}$ (Complement Modus Ponens)**

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

And finally, the following two rules, both of which leave the syntax unchanged, are ND reformulations of Cooper storage and retrieval, respectively.

(34) **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)

(35) **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)

It should be noted that, since co-contexts are sets, not lists, retrieval is nondeterministic not only with respect to which node in the proof tree it takes place at, but also with respect to which of the stored operators is retrieved.

4 Analysis of Quantifier Raising in English

Our English fragment will employ the following lexicon. By convention, for any lexical entry, the words and the semantic constants are presupposed to have already been licensed, respectively, by the syntactic and semantic logics.

(36) **Lexicon for English Fragment**

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

$\vdash$ everyone, everyone' : NP, $\iota^{\pi}_{\pi} \dashv$

$\vdash$ someone, someone' : NP, $\iota^{\pi}_{\pi} \dashv$

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

$\vdash \mathsf{thinks}, \mathsf{think'} : \mathrm{S} \multimap_{\mathrm{C}} \mathrm{NP} \multimap_{\mathrm{S}} \mathrm{S}, \pi \to \iota \to \pi \dashv$

(37) **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')

(38) **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'($\lambda_w(\forall_x(\mathsf{person'}(x)(w) \to \mathsf{like'}(x)(\mathsf{Kim'})(w))))(\mathsf{Chris'})$: s $\to$ t

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

(39) **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.

5 Background for the Analysis of *Wh*-in-Situ

In dealing with the semantics of (possibly multiple) in-situ constituent questions, we take as our target (Ty2) semantics a variant (Pollard 2008c) of Karttunen's (1977) semantics of interrogatives, which analyzes interrogative denotations as sets of propositions. We follow Karttunen in the case of polar questions; but for n-place constituent questions, we take the denotation to be (the curried form of) a function from n-tuples to propositions:[22]

[22] A set of propositions can then be recovered as the *range* of this function. This set differs from the Karttunen semantics in having both positive and negative 'atomic answers' as members. Additionally, our interrogative meanings yield a refinement of the Groenendijk-Stokhof partition semantics by taking the induced equivalence relation on worlds. See Pollard 2008c for detailed discussion.

(40) **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 extension: $(s \to t) \to t$ (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.

(41) **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 extension: $(s \to e) \to (s \to t) \to t$ (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.

(42) **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 extension: $(s \to t) \to (s \to e) \to (s \to t) \to t$ (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.

The fact that not all questions have the same type complicates the analysis of in-situ multiple constituent questions as compared with the analysis of multiple quantifier retrieval (39). For example, scoping one in-situ *wh*-operator at a proposition produces a unary constituent question, so its type must be $\iota_{\pi}^{\kappa_1}$. Thus, if we want to scope a second in-situ *wh*-operator over that unary constituent question to form a binary constituent question, then *its* type must be $\iota_{\kappa_1}^{\kappa_2}$, and so forth. So

unlike QNPs, *wh*-operators must be (in principal infinitely) polymorphic. Note that this polymorphism has nothing to do with the depth of embedding of the *sentences* at which the operator is retrieved, but only with the operator's scoping order (in the sequence of all the wh-operators scoped within a given sentence).

Our analysis will make use of a number of Ty2 logical constants, defined by the following meaning postulates:

(43) **Ty2 Meaning Postulates for Some Useful Logical Constants**

a. $\vdash \mathsf{id}_n = \lambda_Z Z$ (for $Z : \tau(\kappa_n)$)
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. $\vdash \mathsf{equals'}_A = \lambda_x \lambda_y \lambda_w (x = y)$
f. $\vdash \mathsf{whether'} = \lambda_q \lambda_p (p \wedge ((p \ \mathsf{equals'}\ q) \vee (p \ \mathsf{equals'}\ \mathsf{not'}(q))))$
g. $\vdash \mathsf{which}^0 = \lambda_Q \lambda_P \lambda_x \lambda_p (Q(x) \ \mathsf{and'}\ \mathsf{whether'}(P(x))(p))$
h. $\vdash \mathsf{which}^n = \lambda_Q \lambda_Z \lambda_{x_0} \ldots \lambda_{x_n} \lambda_p (Q(x) \ \mathsf{and'}\ Z(x_0) \ldots (x_n)(p))$ $(n > 0)$

The last two are the Ty2 meanings of the interrogative determiner *which*. We do not include determiners in this fragment, but these meanings are used to define the following nonlogical constants:

(44) **Ty2 Meaning Postulates for some Nonlogical Constants**
For $n \in \omega$:

a. $\vdash \mathsf{who}^n = \mathsf{which}^n(\mathsf{person'})$
b. $\vdash \mathsf{what}^n = \mathsf{which}^n(\mathsf{thing'})$

6 Chinese Interrogatives

We turn now to the analysis of so-called covert *wh*-movement in Chinese.[23]

Our Chinese fragment uses the same types, categories, and (semantic, syntactic, and interface) rule schemata as the English, but a different lexicon:

(45) **Lexicon for Chinese Fragment**

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

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

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

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

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

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

[23] The analysis we will propose here improves on an earlier version (Pollard 2007a,b) which required construction-specific rules for different in-situ operators.

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

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

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

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

Note that *xibuxihuan* 'like?' is a partial-reduplicative interrogative verb form, used for forming (both root and embedded) polar questions. The verb *xiangzhidao* 'wonder' has to be type-schematized according to the type of question expressed by the sentential complement. And the *sh*-interrogative words have to be type-schematized according by their scope type (and corresponding result type). This fragment produces analyses such as the following:

(47) **A Simple Chinese Sentence**

a. Zhangsan xihuan Lisi.

b. Zhangsan like Lisi

c. Zhangsan likes Lisi.'

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

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

(48) **A Chinese Polar Question**

a. Zhangsan xi-bu-xihuan Lisi?

b. Zhangsan like? Lisi

c. 'Does Zhangsan like Lisi?'

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

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

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

a. Zhangsan xihuan shenme?

b. Zhangsan like who

c. 'What does Zhangsan like?'

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

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

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

a. Shei xihuan shenme?

b. who like what

c. Who likes what?

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

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

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

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

Finally, we consider so-called Baker-type ambiguities. Baker (1970) noticed that English sentences like the following are ambiguous:

(51) **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.

A full account of thus phenomenon in English depends on an analysis of *overt* movement, which is beyond the scope of this paper (but see Pollard 2008a). Instead, we analyze the corresponding facts of Chinese, which involve only covert movement.

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

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

b. Zhangsan wonder who like what

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

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

(53) **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 (52d), then the root sentence expresses a proposition and the embedded sentence expresses a binary question.

c. If one scopes high and the other low (52e,52f), 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.

7 Conclusion

We have presented a new, simple, and formally precise account of so-called covert movement phenomena. The key ideas of the account are these:

(54) **The Key Ideas Summarized**

- As in CG, both the syntax and the semantics of a linguistic expression are proofs.
- But unlike mainstream CG, the syntax-semantics interface is not a function, so operator-scope ambiguities need not have syntactic reflections.
- Thus the syntax is simple.
- And unlike TG, the interface is not a nondeterministic process made up of sequences of structural operations on trees.
- Instead, it is just a recursive specification of which proof pairs go together (parallel derivational architecture).
- The key insights embodied in the the semantic logic RC go back to the 1970s: Cooper's storage and retrieval.
- The RC formulation generalizes Carpenter's ND rules for Moortgat's $\Uparrow$, but only in the semantic logic (not the syntactic one).
- The transform from RC to TLC is simple.[24]

A number of issues remain to be addressed. For one thing, the relationship between covert and overt movement needs to be clarified. Some preliminary steps in this direction are taken in Pollard 2008b,d. In essence, the approach taken there is to reconstruct the analysis of overt movement in Gazdar 1981, using (abstract) syntactic operators paired with operator meanings of the the same general character as those that occur in the co-context. Such syntactic operators bind a syntactic variable ('trace of overt movement') in a sentence in much the same way that a quantifier retrieved from the co-store binds a semantic variable in a proposition, except that rather then being retrieved, it is just an ordinary logical premiss.

Second, it remains unclear how ultimately to make sense of the co-store, and the storage and retrieval mechanisms, in logical (or categorical) terms. In this connection, de Groote et al. (2009) show that the analysis of covert movement set forth above can be assimilated to the CVG analysis of overt movement just mentioned, provided we analyze an *in situ* operator as an ordinary premiss with an operator type, which, when applied to its 'gappy' sentential argument,

[24] It would be instructive to understand the the connection between this transform and ones employed in many recent CG approaches (e.g. de Groote 2001a, Barker 2002, Shan 2002 and 2004, Moortgat 2007, and Bernardi and Moortgat 2007) based on CPS transforms (Plotkin 1975, Felleisen 1988, Danvy and Filinski 1990, Parigot 1992 and 2000, and Curien and Herbelin 2000).

in effect lowers itself into the trace position via β-reduction.[25] In other words, a CVG with co-store can be algorithmically converted into an ordinary multimodal categorial grammar without co-store, with CVG derivations being globally transformed into ordinary proofs that make no use of storage or retrieval.

This state of affairs is vaguely analogous to CPS transforms that map programs with control operators into pure functional programs. But what is missing is a convincing logical or categorical characterization of the CVG-to-CG transform. In the absence of such a characterization, perhaps the best face we can put onto the the storage-and-retrieval machinery is that it provides a kind of syntactic sugar for linguists with a taste for surface-oriented syntax. To put a more positive spin on it, the de Groote et al.transform can be taken as establishing that Cooper-style storage-and-retrieval machinery actually has a precise meaning (which is given by the transform).

A third, and potentially more serious challenge for the framework presented above is the existence of the linguistic phenomenon of *parasitic scope* discussed by Barker (2007). This has to do with seemingly quantificational expressions whose scope, as Barker puts it, 'depends on the scope of some other scope-taking element in the sentence'. For example, in the following sentences

(55)

a. Anna and Bill read the same book.

b. John hit and killed the same man

the interpretations of the NPs introduced by *the same* depend on the interpretations of the coordinate expressions *Anna and Bill* and *hit and killed*. Barker argues persuasively that such phenomena resist coherent analysis under familiar approaches to quantifer scope, and offers a type-logical analysis that makes use of both continuations and choice functions. Ongoing investigation of parasitic scope (broadly construed to include similar phenomena such as remnant comparatives and internal readings of superlatives) suggest that, although neither continuations nor choice functions are required for the analysis of parasitic scope, a convincing characterization of such constructions in terms of storage-and-retrieval is simply not avaiable.[26] If so, then it may well be that, after 35 years of yeoman service, storage-and-retrieval technology is overdue for retirement.

References

Bach, E. and B. Partee. 1980. Anaphora and semantic structure. in J. Krieman and A. Ojeda, eds., *Papers from the Parasession on Pronouns and Anaphora*. Chicago: Chicago Linguistic Society, pp. 1-28.

[25] This analysis is broadly similar to Oehrle's (1994) simulation of Montagovian 'quantifying in', except that on Oehrle's approach, the β-reduction that effects the 'lowering' is in the concrete (rather than the abstract) syntax.

[26] Unless one is willing to countenance new rule schemas which allow elements within co-stores to interact with other (rather than merely being stored and retrieved).

Anoun, H. and A. Lecomte. 2007. *Linear Grammars with Labels.* In G. Satta, P. Monachesi, G. Penn, and S. Wintner, eds., *Proceedings of Formal Grammar 2006.* Stanford: CSLI.

Aoun, J. and A. Li. 1993. *Wh*-elements in situ: syntax or LF? *Linguistic Inquiry* 24(2):199-238.

Baker, C. 1970. Note on the description of English questions: the role of an abstract question morpheme. *Foundations of Language* 6:197-219.

Barker, C. 2002. Continuations and the nature of quantification. *Natural Language Semantics* 10:211-242.

Barker, C. 2007. Parasitic scope. *Linguistics and Philosophy* 30.4:407-444.

Benton, N., G. Bierman, V. de Paiva, and M. Hyland. 1992. Term assignment for intuitionistic linear logic. Technical Report No. 262, University of Cambridge, August 1992.

Bernardi, R. and M. Moortgat. 2007. Continuation semantics for symmetric categorial grammar. In D. Leivant and R. de Queiroz, eds., pp. 53-71.

Buszkowski, W. 1987. The logic of types. In J. Srzednicki, ed., *Initiatives in Logic.* Dordrecht: M. Nijhoff, pp. 180-206.

Carpenter, B. 1997. *Type-Logical Semantics.* Cambridge, MA and London: MIT Press.

Chomsky, N. 1976. Conditions on rules of grammar. *Linguistic Analysis* 2:303-351.

Chomsky, N. 1977. On *wh*-movement. In P. Culicover, T. Wasow, and A. Akmajian, eds., *Formal Syntax.* New York: Academic Press, pp. 71-132.

Chomsky, N. 1981. *Lectures on Government and Binding.* Dordecht: Foris.

Chomsky, N. 1995 *The Minimalist Program.* Cambridge, MA: MIT Press.

Culicover, P. and R. Jackendoff. 2005. *Simpler Syntax.* Oxford: Oxford University Press.

Curien, P.-L., and H. Herbelin. 2000. The duality of computation. In ICFP'00, pp. 233-243.

Cooper, R. 1975. *Montague's Semantic Theory and Transformational Syntax.* Ph.D. dissertation, University of Massachusetts at Amherst.

Cooper, R. 1983. *Quantification and Syntactic Theory.* Dordrecht: Reidel.

Copestake, A., D. Flickinger, C. Pollard, and I. Sag. 2005. Minimal recursion semantics: an introduction. *Journal for Research on Language and Computation* 3.4:281-332.

Curry, H. and R. Feys. 1958. *Combinatory Logic, Vol. 1.* Amsterdam: North-Holland.

Danvy, O. and A. Filinski. 1990. Abstracting control. In *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming.* New York: ACM Press, pp. 151-160.

de Groote, P. 2001a. Type raising, continuations, and classical logic. In R. van Rooy and M. Stokhof, eds., *Proceedings of the Thirteenth Amsterdam Colloquium.* Amsterdam: Institute for Logic, Language, and Computation, Universiteit van Amsterdam, pp. 97-101.

de Groote, P. 2001b. Toward abstract categorial grammars. In EACL 10:148-155.

de Groote, P., S. Pogodalla, and C. Pollard. In press. About parallel and syntactocentric formalisms: what the encoding of Convergent Grammar into Abstract Categorial Grammar tells us. To appear in *Fundamenta Informaticae.*

Felleisen, M. 1988. The theory and practice of first-class prompts. In POPL'88. New York: ACM Press, pp. 180-190.

Gabbay, D. and R. de Queiroz. 1992. Extending the Curry-Howard interpretation to linear, relevant, and other resource logics. *Journal of Symbolic Logic* 57(4):1319-1365.

Gazdar, G. 1981. Unbounded dependencies and coordinate structure. *Linguistic Inquiry* 12:155-184.

Gallin, D. 1975. *Intensional and Higher Order Modal Logic*. Amsterdam: North-Holland.

Groenendijk, J. and M. Stokhof. 1984. *Studies on the Semantics of Questions and the Pragmatics of Answers*. Ph. D. dissertation, University of Amsterdam.

Hendriks, H. 1993. *Studied Flexibility: Categories and Types in Syntax and Semantics*. Ph.D. dissertation, Universiteit van Amsterdam.

Howard, W. 1980. [1969] The formulas-as-types notion of construction. In J. Hindley and J. Seldin, eds., *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*. New York: Academic Press.

Huang, C.-T. James. 1992. *Logical Relations in Chinese and the Theory of Grammar*. Ph.D. dissertation, MIT.

Jacobson, P. 1999. Toward a variable-free semantics. *Linguistics and Philosophy* 22(2): 117-184.

Jay, B. 1989. Languages for monoidal categories. *Journal of Pure and Applied Algebra* 59:61-85.

Karttunen, L. 1977. Syntax and semantics of questions. *Linguistics and Philosophy 1*, 3-44.

Lambek, J. 1958. The mathematics of sentence structure. *American Mathematical Monthly* 65:363-386.

Lambek, J. 1988. Categorial and categorical grammars. In R. Oehrle, E. Bach, and D. Wheeler, eds, *Categorial Grammars and Natural Language Structures*. Dordrecht: Reidel, pp. 297-317.

Lecomte, A. 2005. Categorial grammar for minimalism. In C. Casdio, P. Scott, and R. Seely, eds., *Language and Grammar: Studies in Mathematical Linguistics and Natural Language*. Stanford: CSLI, pp. 163-188.

Lecomte, A. and C. Retoré. 2002. Bi-grammars: a logical system for syntax, semantics, and their correspondence. Unpublished paper presented at Formal Grammar 2002 (Trento).

Leivant, D. and R. de Queiroz, eds. 2007. *Proceedings of the Fourteenth Workshop on Logic, Language, Information, and Computation (WoLLIC 2008)*. Vol. 4576 of *Lecture Notes in Computer Science*. Berlin and Heidelberg: Springer.

Mackie, I., L. Román, and S. Abramsky. 1993. An internal language for autonomous categories. *Proceedings of the First Imperial College Department of Computing Workshop on Theory and Formal Methods*. London: Springer, pp. 235-246.

May, R. 1977. *The grammar of quantification*. Ph.D. dissertation, MIT. Reprinted by Garland, New York, 1991.

May, R. 1985. *Logical Form: Its Structure and Derivation*. Cambridge, MA: MIT Press.

Mints, G. 1981 [1977] Closed categories and the theory of proofs. *Journal of Soviet Mathematics* 15(1)45-62. Translated from *Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Insitituta* im. V.A. Steklova AN SSSR, Vol. 68:83-114 (1977).

Mitchell, J. and P. Scott. 1989. Typed lambda models and cartesian closed categories. *Contemporary Mathematics* 92:301-316.

Montague, R. 1974. [1970] The proper treatment of quantification in ordinary English. In R. Thomason, ed., *Formal Philosophy: Selected Papers of Richard Montague*. New Haven: Yale University Press, pp. 247-270. Originally published in J. Hintikka, J. Moravcsik, and P. Suppes, eds., *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics* Dordrecht: Reidel.

Moortgat, M. 1988. *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus*. Ph.D. dissertation, Universiteit van Amsterdam. Dordrecht: Foris.

Moortgat, M. 1996. Generalized quantifiers and discontinuous type constructors. In H. Bunt and A. van Horck, eds., *Discontinuous Constituency*. De Gruyter, pp. 181-207.

Moortgat, M. 1997. Categorial type logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*, chapter 2. Elsevier/MIT Press, pp. 93-177.

Moortgat, M. 2007. Symmetries in natural language syntax and semantics. In D. Leivant and R. de Queiroz, eds., pp. 264-268.

Morrill, G. 1994. *Type Logical Grammar: Categorial Logic of Signs*. Dordrecht: Kluwer.

Muskens, R. 2003. Languages, lambdas, and logic. In G.-J. Kruiff and R. Oehrle, eds., *Resource Sensitivity in Binding and Anaphora*. Kluwer., pp. 23-54.

Oehrle, R. 1994. Term-labelled categorial type systems. *Linguistics and Philosophy* 17:633-678.

Parigot, M. 1992. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR)*, St. Petersburg. LNCS 624.

Parigot, M. 2000. On the computational interpretation of negation. In P. Clote and H. Schwichtenberg, eds., *Computer Science Logic*, vol. 1862 of *Lecture Notes in Computer Science*. Berlin and Heidelberg: Springer, pp. 472-484.

Pesetsky, D. 1987. Wh-in-situ: movement and unselective binding. In E. Reuland and A. ter Meulen, eds., *The Representation of (In-) Definiteness*. Cambridge, MA: MIT Pres, pp. 98-129.

Plotkin, G. 1975. Call-by-name, call-by-value, and the λ-calculus. *Theoretical Computer Science* 1(2):125-159.

Pollard, C. and I. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press, and Stanford: CSLI.

Pollard, C. 2004. Type-logical HPSG. In *Proceedings of Formal Grammar 2004*. `http://www.ling.ohio-state.edu/~hana/hog/pollard2004-FG.pdf`.

Pollard, C. 2007a. The logics of overt and covert movement in a relational type-theoretic grammar. Presented at the Fourth Workshop on Lambda Calculus and Formal Grammar, LORIA, Nancy, Sept. 2007.
`http://www.ling.ohio-state.edu/~hana/hog/pollard2007-lcfg.pdf`.

Pollard, C. 2007b. The logic of pied piping. Presented at Colloque en l'honneur d'Alain Lecomte, Pauillac, Nov. 2007.
`http://www.ling.ohio-state.edu/~hana/hog/pollard2007-pied.pdf`.

Pollard, C. 2008a. The calculus of responsibility and commitment. Unpublished paper accepted for presentation at the Workshop on Ludics, Dialog, and Interaction, Autrans, France, May 2008. `http://www.ling.ohio-state.edu/~pollard/cvg/autrans.pdf`.

Pollard, C. 2008b. A parallel derivational architecture for the syntax-semantics interface. Presented at the ESSLLI 2008 Workshop on What Syntax Feeds Semantics, Hamburg, August 2008.
http://www.ling.ohio-state.edu/∼pollard/cvg/para-slides.pdf.

Pollard, C. 2008c. What do interrogative sentences refer to? Unpublished paper, Ohio State University, Universitat Rovira i Virgili, and INRIA-Lorraine. Presented at the Workshop on Reference to Abstract Objects, Universitat Pompeu Fabra, Barcelona, March 2008. http://www.ling.ohio-state.edu/∼pollard/cvg/absobj.pdf.

Pollard, C. 2008d. Convergent Grammar. Slides for ESSLLI 2008 course, Hamburg, August 2008. http://www.ling.ohio-state.edu/∼pollard/cvg/day2.pdf.

Pollard, C. 2008d. Hyperintensions. *Journal of Logic and Computation* 18.2:257-282. http://www.ling.ohio-state.edu/∼pollard/cvg/hyper.pdf.

Pollard, C. 2008e. Hyperintensional questions. In W. Hodges and R. de Queiroz, eds., *Proceedings of the 15th Annual Workshop on Logic, Language, Information, and Computation (WoLLIC 08): Heriot-Watt University 2008*. Springer Lecture Notes in Artificial Intelligence 5110::261-274.
http://www.ling.ohio-state.edu/∼pollard/cvg/wollic08.pdf.

Ranta, A. 2002. Grammatical Framework. *Journal of Functional Programming* 14:145-189.

Shan, C.-c. 2002. A continuation semantics of interrogatives that accounts for Baker's ambiguity. In B. Jackson, ed., *Proceedings of Semantics and Linguistic Theory 12*. Ithaca: Cornell University Press, pp. 246-265.

Shan, C.-c. 2004. Delimited continuations in natural language: quantification and polar sensitivity. *Continuation Workshop 2004*, Venice.

Steedman, M. 1996. *Surface Structure and Interpretation*. Cambridge, MA: MIT Press.

van Benthem, J. 1983. The semantics of variety in categorial grammar. Report 83-29, Department of Mathematics, Simon Fraser University, Vancouver. Reprinted in W. Buszkowski, J. van Benthem, and W. Marciszewski, eds., *Categorial Grammar*. Amsterdam: John Benjamins, pp. 37-55.

Wansing, H. 1992. 'Formulas-as-types' for a hierarchy of sublogics of intuitionistic proposiitonal logic. In D. Pearce and H. Wansing, eds., *Nonclassical Logics and Information Processing*. LNAI 619. Berlin: Springer, pp. 125-145.