8

Type-Logical HPSG

Carl Pollard

8.1 Introduction

Pullum and Scholz (2001) bifurcate 20th-century syntactic research frameworks into two principal paradigms: *model-theoretic syntax* (MTS, e.g. arc pair grammar, construction grammar, and HPSG) and *generative-enumerative syntax* (GES, e.g. transformational grammar and categorial grammar, including type-logical grammar (TLG)). Pullum and Scholz argue on empirical grounds for the superiority of MTS over GES. Although I think their arguments are vulnerable to criticism on a number of counts, my purpose here is not to criticize MTS but rather to argue that one need not choose between MTS and GES. More precisely, I propose a framework, *higher-order grammar* (hereafter HOG) which at once embodies both model-theoretic and proof-theoretic aspects. To put it another way, HOG is in the intersection of the MTS and GES paradigms; its MTS and GES aspects are not in competition, but rather complementary.

The comparison between TLG (Morrill, 1994, Moortgat, 1997) and HPSG (Pollard and Sag, 1994) is of particular interest because, among all the widely employed syntactic frameworks, they have been especially committed to explicit formalization of linguistic theory within logic. Given this shared concern with formal precision (to say nothing of other commonalities such as lexicocentrism and concern with computational tractability), one might well wonder why the research communities associated with these two frameworks have not merged into a single community. The principal scientific reason for the separa-

Proceedings of Formal Grammar 2004.
Paola Monachesi, Gerhard Jäger,
Gerald Penn and Shuly Wintner (eds.).

tion is that the logical foundations of the two frameworks are seemingly incompatible.

Leaving aside for the moment semantic interpretation, in TLG words (thought of as prosodic/phonological entities) are assigned to types—formulas in a resource-sensitive logic (usually an elaboration of Lambek's 1958, 1961 syntactic calculus) and then the assignment is extended to word strings by well-known proof-theoretic means. In HPSG, by contrast, one starts with a set of "candidate" structures (in the terminology of Carpenter (1992), the totally well-typed, sort-resolved, inequation-resolved feature structures over a given signature of sorts and features), and then discards the ones that fail to satisfy the grammar, a set of axioms (grammatical constraints) written in a (quite idiosyncratic) classical propositional logic, viz. RSRL (Richter, 2000). TLG, then, is a GES framework because the derivations are (at least) recursively enumerable, whereas HPSG is a MTS framework because the well-formed structures are models of the logical theory axiomatized by the constraints. It would appear, then, that the logics underlying the two frameworks bear no interesting relationship. But as will be shown presently, linguistic type logics and linguistic constraint logics can be seen to be intimately related, if viewed from a higher-order perspective.

HOG is an outgrowth of research over the past several years aimed at solving some of HPSG's foundational problems, which mostly arise from the absence of functional types. Once we opt for a classical logic with functional types for expressing linguistic constraints, some form or other of higher-order logic (HOL) naturally suggests itself. In fact, the use HOL for linguistic description has been advocated by others (e.g., Moshier, 1999, Ranta, In press, Penn and Hoetmer, 2003). The present proposal, however, is unique in two respects. First, HOG employs a single HOL for all three of syntax, semantics, and phonology (as well as the syntax-semantics and syntax-phonology interfaces). Correspondingly, as in HPSG, syntactic, semantic, and phonological entities inhabit a model of a grammar. And second, HOG exploits the formal parallel between, on the one hand, the cartesian type constructors (product ($\times$) and exponential ($\Rightarrow$)) of the typed lambda calculus that underly HOL and, on the other hand, the tensor type constructors (tensor product and directional slashes) of the Lambek calculus, thereby enabling analogs of linguistic analyses originating within TLG to be developed in an MTS setting. (A third difference, namely that HOG is *categorical* in the sense that the types and entities of a given natural language are construed, respectively, as objects and arrows of a category, with phonological and semantic intepretation as endofunctors, is discussed in (Pollard, 2004).)

8.2 HOG and HPSG

To clarify the connection with HPSG, a HOG can be characterized roughly as follows. (1) The grammar is is written in a classical HOL rather than in RSRL. Thus HOG is free of the idiosyncrasies of RSRL such as chains (Richter, 2000) and the concomitant undecidability of finite model-checking (Kepser, 2001). (2) The types of the HOL replace HPSG's features structure types. More specifically, feature structure types become indexed product types (Barr and Wells, 1999) HPSG partitions of a type are expressed as coproducts (model-theoretically, disjoint unions), and types SET-OF[A] are realized as functional types $A \Rightarrow$ Bool. (3) A sign is modelled not by a feature structure but rather by the denotation (in a model of the grammar) of a closed term. (4) Unlike HPSG, HOG does not have a type Sign; instead signs are of many types (namely the ones where the interpretive endofunctors are defined). (5) "Unsaturated" signs (in HPSG terms, signs with non-null valence features) have functional types. Hence there are no valence features, so the perennial problem of how to instantiate undischarged valence features (e.g. the subject of the infinitive in *to err is human*) to satisfy total well-typedness does not arise. (6) Semantic interpretation, treated in HPSG as just another feature (CONTENT) of signs, is treated in HOG as a (schematic polymorphic) function from signs to semantic entities, that is, a type-indexed family of functions each of whose domains is one of the sign types. Since the HOL is already a lambda calculus, there is no need to encode Ty2 terms as features structures or lambda conversion as an RSRL relation (Richter and Sailer, 2003). (7) Analogously, phonological interpretation, also treated in HPSG as a feature (PHONOLOGY) of signs, is a (schematic polymorphic) function from signs to phonological entities. Constraints on phonological entities (e.g. phonotactic constraints) can then be expressed directly as nonlogical axioms of the grammar. (8) Since there is just equality *simpliciter* rather than a distinction between type identity and token identity, the framework is free of Höhle's Problem (that a sentence containing two occurrences of the same sign is spuriously ambiguous as to whether the occurrences are type-identical or merely token identical). Some of these these points will be elaborated below; others are discussed elsewhere (as specified).

8.3 HOG resolves the MTS-GES dichotomy

The HOG architecture provides insight into the relationship between TLG's type logic and HPSG's constraint logic, since it has analogs of both. The HOG analog of TLG's type logic is just the HOL's type

system, which, as for any typed lambda calculus ((Curry and Feys, 1958, Howard, 1980) forms an intuitionistic propositional logic with the type constructors as the logical connectives (though not a resource-sensitive one as in TLG). And the HOG analog of HPSG's constraint logic (RSRL) is just the higher-order logic of terms: both are quantificational logics with all the familiar boolean connectives, and both are used to impose well-formedness constraints on linguistic entities. TLG, however, lacks an analogue of the constraint logic. In HPSG, on the other hand, what is missing is the type logic. But in HOG there is both a type logic and a constraint logic, and the latter is the proof term calculus of the former. Thus any grammar will be a theory written in the HOL of choice, and in a model of that theory, any entity of a given type will satisfy all the constraints that the grammar imposes on entities of that type. In this respect HOG is an MTS framework (like HPSG). But at the same time, any one of the family of equivalent terms denoting that entity encodes (as per Curry-Howard) a natural-deduction proof of its type. So as long as grammars are written in such a way to ensure that the set of signs of type S (which is in one-to-one correspondence with the set of normalized proofs of type S) is recursively enumerable, HOG is also a GES framework.

8.4 The Logic

HOL (with two types, here called Bool (truth values) and Ent (entities), and the single type constructor $\Rightarrow$), was first placed on a firm footing in the form of Church's 1940 simple theory of types (STT), which moved the term equivalence of the simply-typed lambda calculus into the object language and added constants to serve as logical connectives and quantifiers. Henkin (1950) reaxiomatized STT, added the axiom of propositional extensionality, and proved completeness with respect to the class of models which now bear his name, viz. general Henkin models (here 'general' means that there need only be enough functions to interpret all closed functional terms). Gallin (1975) showed that Ty2 (obtained by adding a type World to Henkin's HOL) was equivalent in a clearly defined sense to Montague's intensional logic IL (equipped with a suitable proof theory).

As pleasant to work with (and familiar to linguists) as Ty2 is, it is somewhat too blunt an instrument to serve as a general linguistic formalism, even with the addition of arbitrarily many basic types. Experimentation over the past several years points to the need for the following features absent in Ty2:

Indexed Products. The (cartesian) product type constructor $\times$ (conjunction, in terms of the type logic), together with corresponding projection terms and pairing term constructor, is a standard feature of many HOLs and functional programming languges. This makes currying of functions an option rather than a necessity. Even better is the addition of *indexed* products, which allow the factors in product types to be indexed by arbitrary sets of labels (feature names) rather than just by natural numbers (so that the indexed projection functions are the features). Indexed products are a more standard way of doing what linguists do with feature structures.

Coproducts. The (cartesian) coproduct type constructor + (disjunction in the type logic, with corresponding injection functions and copairing term constructor) is interpreted as disjoint union in the models and is therefore ideally suited for partitioning types, as in HPSG type hierarchies.

The addition of product and coproduct (including nullary 1 and 0 respectively) to the original exponential ($\Rightarrow$) constructor makes the type logic into a full intutionistic propositional logic.

Separation types. Separation types (so-called by analogy with the set-theoretic axiom of separation), are subtypes defined by restricting a given type by an open boolean term, e.g. given S as a primitive type, we can define the subtype of finite sentences as the separation type

$$\text{Sfin} =_{def} \ [x \in \text{S} |\ \textsc{Vform}(x) = \ \mathsf{fin}]$$

The (separation) subtypes of any given type should form a boolean algebra. For example, to implement the now-standard analysis of case syncretism (Bayer and Johnson, 1995), given types NPacc and NPgen, we would like to define the type of syncretic accusative-genitive noun phrases as

$$\text{NPnom\_acc} =_{def} \ [x \in \text{NP} \mid \textsc{Case}(x) = \ \mathsf{nom\_acc}] = \text{NPnom} \cap \text{NPacc}$$

Natural Number Type. Incorporation of a natural number type Nat is another standard feature, which in effect builds an analog of the set-theoretic axiom of infinity into the logic and, inter alia, makes the Kleene-* (string) type constructor definable. That is, for each type A there is a type A^* with the expected behavior of strings (see (Pollard and Hana, 2003) for linguistic motivation and application to the analysis of coordination).

Schematic polymorphism. From its inception HPSG has employed, at least informally, some notion of parametric polymorphism (e.g. for

sets or lists all of whose members are to be of the same type A). In HOG we already have strings (A^*) and sets ($A \Rightarrow$ Bool), but there is still a need for limited polymorphism, e.g to define the semantic and phonological interpretation functions across the kind of sign types. Experience thus far suggests that schematic (or abbreviatory) polymorphism is sufficient; that is, no new types (or quantification over types) are introduced, but a family of functions can be defined schematically across a family of types (here, the sign types).

The higher-order categorical logic of Lambek and Scott (1986) provides a good point of departure for satisfying the foregoing desiderata: it has products, (definable) coproducts, natural number type, and (separation) subtypes. Moreover the subtypes of a given type form a Heyting algebra, which becomes boolean once the boolean axiom is imposed. This is still a bit too general, because the type of truth values (usually called Ω) can be an arbitrary Boolean algebra (so there can be truth values other than true and false); so in order to get bivalence it is also necessary to impose the type identity $\Omega = 1 + 1$, with Ω being the truth value type and true and false being the canonical injections. (This then justifies the renaming of Ω to Bool.) The models of the resulting logic are categories (abstract mathematical universes) called *bivalent boolean toposes* abstract models of typed lambda calculus with enough additional structure to interpret a propositional type and associated logical constants. Henkin models (when so augmented) are special cases of these. (Readers unfamiliar with category theory can just think of Henkin models augmented with cartesian products and lambda-definable subtypes without being led seriously astray.)

8.5 Syntax

A higher-order grammar is given by specifying three things: (1) the basic types; (2) the basic nonlogical constants (including their types); and (3) the constraints (nonlogical axioms).

Basic types. For purposes of discussion we present a HOG for a simple fragment of English (with noun and verb as the only parts of speech), starting with the following basic types: Phon (phonemes); S (sentences); NP (noun phrases, for the moment limited to nonquantificational ones); N (common noun(phrase)s, setting aside the the question of whether N should be analyzed as the head of NP); Prop (propositions, the semantic interpretations of declarative sentences); Ind (individual concepts, the semantic interpretations of noun phrases); and Ent (entities, the kinds of things that can be the extensions of individual concepts). (Note that the type Bool of truth values, the extensions

of propositions, is already supplied by the logic.) For notational convenience we also provide types for the values of what are treated in HPSG as nonboolean head features in HPSG, such as Vform (verb inflected form), Case (case), and Agr (noun agreement).

Basic constants. Next, we add the basic nonlogical constants. For example, to say that nom is a case value we include in the grammar the basic constant $\mathsf{nom} : 1 \to \text{Case}$, which for familiarity we write as

$\mathsf{nom} \in \text{Case}$

This means that in a model, nom is interpreted as a member of the set that interprets the type symbol Case (more precisely, as a function whose codomain is that set and whose domain is the singleton set $\{0\}$).

Next we add functions which play the same role in HOG that head features play in HPSG, e.g.

$\textsc{Case} \in (\text{NP} \Rightarrow \text{Case})$
$\textsc{Agr} \in (\text{NP} \Rightarrow \text{Agr})$
$\textsc{Vform} \in (\text{S} \Rightarrow \text{Vform})$
$\textsc{Aux} \in (\text{S} \Rightarrow \text{Bool})$
$\textsc{Inv} \in (\text{S} \Rightarrow \text{Bool})$

We turn next to the specification of the lexicon. For example, to include the word *she* as a nominative third-singular-feminine nominative noun, we add the specification

$\mathsf{she} \in \text{NPnom/3fs}$
where the target type is defined as follows:
$\text{NPnom/3fs} =_{def} \ [x \in \text{NP} \mid \textsc{Case}(x) = \mathsf{nom} \wedge \textsc{Agr}(x) = \mathsf{3fs}]$

Note that the definition does not bring the defined type into existence! Its existence is a consequence of the existence of the basic type NP, together with the subtyping provided by the logic; the definition merely provides a handy abbreviation. It is important to be aware that the entity that interprets the constant she is to be thought of as modelling the word *she* qua syntactic entity; it is not a phonological entity. (So far we have said nothing about what the syntactic word she sounds like). The view of signs (syntactic words and phrases) as inhabitants of syntactic types originates with Lambek's 1988, 1999 categorical view of his own syntactic calculus. The principal difference between Lambek's approach and the one advocated here is that they employ different type logics: Lambek calculus vs. intuitionistic propositional logic.

Next we add a couple of finite third-singular verbs to the lexicon (the definitions of the various subtypes of S and NP employed should be obvious):

$\mathsf{swims} \in \text{VP3s/main} =_{def} \ (\text{NPnom/3s} \Rightarrow \text{Sfin/main})$

$\mathsf{sees} \in \text{TVP3s/main} =_{def} \ (\text{NP} \Rightarrow (\text{NPnom/3s} \Rightarrow \text{Sfin/main}))$

These lexical items parallel lexical type assignments in TLG, but there are (at least) three important differences. First, the product and exponential type constructors involved are cartesian, not tensor. Second, there is no mention of phonology (no pairing of word strings with types). And third, words (and phrases) actually inhabit their types, rather than just being assigned to them. In the model, this means that, e.g., the word *swims* (i.e. the interpretation of the constant swims) is a member of the set of third-singular main (i.e. nonauxiliary) verb(phrase)s; in Curry-Howard terms, it means that swims encodes a (one-line) derivation (proof) of the formula VP3s/main.

Note that the lexical entry for *sees* above assigns it a curried type. But we could just as well have given the lexical entry as

$\mathsf{sees}^{\dagger} \in ((\text{NP} \times \text{NPnom/3s}) \Rightarrow \text{Sfin/main})$

where the antecedent type is now suggestive of an HPSG Subcat list. In fact, because of the adjoint relationship between $\times$ and $\Rightarrow$, either lexical entry implies the existence of the other:

$\mathsf{sees}^{\dagger} = \mathsf{uncurry}(\mathsf{sees})$
$\mathsf{sees} = \mathsf{curry}(\mathsf{sees}^{\dagger})$;

thus whether lexical entries are curried or uncurried is strictly a matter of convenience.

In HPSG (as in relational grammar and lexical-functional grammar), grammatical functions (such as subject and complement) are taken as theoretical primitives rather than defined (as has usually been done in categorial grammar) in terms of order of functional application. In HOG, primitive grammatical functions are naturally implemented as *contrafeatures*, i.e. indices of indexed product types that occur as the antecedent in an implicative type:

$\text{TVP3s/main} =_{def} \ ((\textsc{Comp} : \text{NP}, \textsc{Subj} : \text{NPnom/3s}) \Rightarrow \text{Sfin/main})$

This development is discussed further in the full paper; for now we just mention that by using natural generalizations of currying and application to the case of indexed products, it is easy to show that the analogs of standard HPSG constraints (specifically the Head Feature Principle and the Valence Principle) are just instances of *modus ponens* with respect to the type logic. Suitably refined, this technique is also applicable to constructors corresponding to HPSG features for handling various types of unbounded dependencies such as Slash (for "*wh*-movement" gaps, including parasitic gaps), Rel (for pied-piped relative pronouns), and Qstore (for unscoped quantificational NPs), etc.

To illustrate, a simple example is provided of how phrasal signs come about (ignoring morphosyntactic features in order to simplify the exposition). Assuming we are given the three lexical entries $\mathsf{kim} \in \mathrm{NP}$, $\mathsf{sandy} \in \mathrm{NP}$, and $\mathsf{sees} \in \mathrm{TVP}$, we can form the term $\mathsf{sees}(\mathsf{sandy})(\mathsf{kim}) \in \mathrm{S}$ by successive functional application. In a model, this term denotes a certain sentence (i.e. member of the set that interprets the type S). This sentence will be mapped by the semantic interpretation functor to a certain proposition (member of the set that interprets the type Prop), and by the phonological interpretation functor to a certain string of phonological words (member of the set that interprets the type $\mathrm{Phoneme}^{**}$). In terms of the type logic, this term corresponds to a certain intuitionistic proof that uses *modus ponens* twice to prove the atomic formula S from the premises NP, NP, and $\mathrm{NP} \Rightarrow (\mathrm{NP} \Rightarrow \mathrm{S})$. Thus the relationship between the derivation of the sentence and the sentence itself is that, literally, the latter is the model-theoretic interpretation of the former. Thus, in the HOG setting, the Curry-Howard isomorphism resolves the distinction between type-logical and constraint-based grammar.

Syntactic constraints (nonlogical axioms). As noted above, some well-established HPSG constraints, such as the Head Feature Principle and the Valence Principle, whose essential purpose is to simulate functional application, come for free. Others, such as the constraints that govern the "discharge" of Nonlocal features, can presumably be absorbed into the general machinery for handling the corresponding type constructors (as is done in TLG), but others may have to be stated as nonlogical axioms, e.g. the English constraints on nested dependencies, which differ from (say) the Swedish ones; constraints on the distribution of parasitic gaps; the constraint that Que-binding is possible at infinite VP or finite S but nowhere else; the pan-Germanic (but not pan-Slavic) constraint that Slash-binders (or *Vorfeld* occupants) must be "constituents" (i.e. cannot be of cartesian product types).

One type of syntactic constraint that is straightforwardly dealt with in HOG is feature co-occurrence restrictions. For example the contraint that in English inverted sentences must be headed by a finite auxiliary can be expresseed as a nonlogical axiom:

$$\forall x(\textsc{Inv}(x) \Rightarrow (\textsc{Aux}(x) \wedge \textsc{Vform}(x) = \mathsf{fin}))$$

where x is a variable of type S. To take another example, consider the hypothesis that, in Polish, nominative and accusative case never syncretize. This can be expressed by the nonlogical axiom

$$\neg\exists x(x = x)$$

where x is a variable of type NPnom_acc.

8.6 Semantics

The HOG approach to semantic interpretation follows up a suggestion due to (Montague, 1974, 263). Recall that in PTQ, translation is treated as a relation between English expressions (in the sense of strings of basic expressions) and terms of Montague's intensional logic IL. Montague's suggestion is to revise the grammar architecture so translation becomes a function from *derivations* (PTQ analysis trees) to IL terms. But our sign-denoting terms can be thought of as encoding proofs, which are analogous to PTQ-style analysis trees; so we implement Montague's suggestion by treating semantic interpretation as a *translation* from syntactic terms to semantic terms. Following Lambek and Scott (1986), here *translation* means that: (1) each sign type translates to a semantic type; (2) cartesian products translate to cartesian products; (3) basic terms of a given sign type translate to closed terms of the corresponding semantic type; (4) the translation extends uniquely to all terms by translating lambda abstraction to lambda abstraction, application to application, and pairing to pairing. In short, translation preserves all lambda-calculus constructs. Additionally, semantic interpretation is required to be a *logical* translation (i.e. to preserve all logical constants). This is a very strong hypothesis about the nature of the syntax-semantics interface, and one that is not easily expressible in HPSG.

The details of the semantic translation are discussed elsewhere. For now we just note that the proposed semantics is *hyperintensional* in the sense of being finer-grained than the usual intensional semantics; i.e. two signs can have interpretations whose denotations coincide in every world, yet are distinct. The trick is to take propositions as primitive, and entailment as (an appropriately axiomatized) preorder on propositions (i.e. a constant of type $(\mathrm{Prop} \times \mathrm{Prop}) \Rightarrow \mathrm{Bool}$) and then use subtyping to *define* the type of worlds as the type of all subsets of the set of propositions which are ultrafilters (maximal consistent sets) relative to the entailment preorder. The hyperintensionality is a consequence of the fact that entailment is only a preorder, not an order (so e.g. two distinct propositions can entail each other). See (Pollard, in preparation) for details.

The upshot is that semantic interpretation is a (schematic polymorphic) function sem_A whose domain is the sign types, with NP and S mapping to Ind and Prop (propositions) respectively. The semantic types for the translations of signs belonging to nonbasic sign types is then determined by the requirement that semantic interpretation be a logical translation (as described above). The semantic interpretations

of lexical signs are assigned by constraints such as:

$\mathsf{sem(kim)} = \mathsf{kim}'$
$\mathsf{sem(sandy)} = \mathsf{sandy}'$
$\mathsf{sem(sees)} = \mathsf{see}'$

which in concert with the logical translation condition, uniquely determines semantic interpretation for all signs. For example:

$\mathsf{sem(sees(sandy)(kim))} = \mathsf{(sem(sees))(sem(sandy))(sem(kim))} =$
$= \mathsf{see}'(\mathsf{sandy}')(\mathsf{kim}')$

8.7 Phonology

HOG phonology can be summarized in one sentence: phonological interpretation, like semantic interpretation, is a logical translation. This entails, inter alia, the following things: (1) It is impossible to tell by looking at (the term denoting) a sign what it sounds like. (2) Phonological entities are in the model and denoted by lambda terms. (3) Phonological interpretation is a translation from terms that denote signs to terms that denote phonological entities. (4) Such a translation is specified by defining it on lexical signs; the extension to phrases is uniquely determined by the requirment that phonological interpretation be a logical translation. (5) The HOL can be used to express phonotactic constraints.

This may sound like a radical program for phonology, but a good deal of it is historically grounded. The first point is closely connected with Curry's 1961 version of type-logical syntax, which insisted on a clean separation between abstract syntactic combinatorics (in Curry's term, *tectogrammar*), and the concrete realizations of syntactic entities (*phenogrammar*); in fact, Curry faulted Lambek's calculus for failing to maintain this distinction. The second point is prefigured in categorial phonology (Wheeler, 1981). The third point has a precursor in (Oehrle, 1994) (however, Oehrle's language of phonological terms did not form a logic).

The following sketch of HOG phonology is necessarily simplified and limited to segmental phonology. Our point of departure is the basic type Phoneme, so that phonological words have type Phonword $=_{def}$ Phoneme* and the phonological interpretations of syntactically saturated signs are strings of phonological words (type Phonword*). Phonological features for phonemes can be handled formally on a par with head features for saturated signs, and natural classes can be defined as subtypes of the type Phoneme. Phonotactic constraints can be expressed as nonlogical axioms, but first the phonological ontology has to

be enriched to include the kinds of entities to be constrained (e.g. syllables).

As with semantic interpretation, the value of the phonological interpretation functor phon on signs is uniquely determined by the values on the lexical signs (which are specified by grammatical constraints) in concert with the condition that phonological interpretation be a logical translation. As noted above, for the saturated sign types NP and S, the corresponding phonological type is Phonword*, and so, for example, the phonological interpretation of a sign of type VP $=_{def}$ (NP $\Rightarrow$ S) (disregarding coordinate structures) is of type Phonword* $\Rightarrow$ Phonword*, and the phonological interpretation of a sign of type TVP is of type Phonword* $\Rightarrow$ (Phonword* $\Rightarrow$ Phonword*).

A few examples should suffice to make this concrete. To enhance readability, we omit the type subscript on polymophically typed constants. Additionally, we employ the standard notational abuse whereby what should denote a phonological word actually denotes a string of phonological words of length one, e.g. /kɪm/ instead of ⟨/kɪm/⟩; in fact, we compound the abuse by writing, e.g. /siz, kɪm/ instead of ⟨/siz/, /kɪm/⟩. Also, e_A denotes the null A-string and $\overset{\wedge}{}_A$ denotes the polymorphically typed concatenation operator

$\overset{\wedge}{}_A \in (A^* \Rightarrow (A^* \Rightarrow A^*))$

These are subject to the following (type-schematized) monoid constraints (with the variables all of type A):

$\forall x(\mathsf{e}\hat{}x = x)$
$\forall x(x\hat{}\mathsf{e} = x)$
$\forall x, y, z((x\hat{}y)\hat{}z = x\hat{}(y\hat{}z))$

Note that these are nonlogical axioms of the grammar, not a metalinguistically imposed term equivalence as in (Oehrle, 1994).

Phonological interpretations are assigned to lexical signs by nonlogical axioms such as the following:

$\mathsf{phon}(\mathsf{kim}) =$ /kɪm/
$\mathsf{phon}(\mathsf{sandy}) =$ /sændi/
$\mathsf{phon}(\mathsf{sees}) = \lambda x \lambda y.y\hat{}$/siz/$\hat{}x$

Just as with semantic interpretation, phonological interpretation of nonlexical signs is uniquely determined by logical functoriality. For example:

$\mathsf{phon}(\mathsf{sees}(\mathsf{sandy})(\mathsf{kim})) = (\mathsf{phon}(\mathsf{sees}))(\mathsf{phon}(\mathsf{sandy}))(\mathsf{phon}(\mathsf{kim})) =$ $(\lambda x \lambda y.y\hat{}$/siz/$\hat{}x)$(/sændi/)(/kɪm/) = /kɪm, siz, sændi/

Note that the lexical entry for *sees* ensures that the first and second syntactic arguments (object and subject respectively) are phonologi-

cally realized to the right and to the left of /siz/. This is why there is no need to split the $\Rightarrow$ constructor into $\backslash$ and /: the directionality of combination is moved out of the syntax and into the phonology (and its interface with syntax). More generally, the resource sensitivity of language is relocated from syntax (where TLG has it) into the phonology; thus the syntactic type logic is not a Lambek calculus but just an ordinary intuitionistic propositional logic with all three of the structural rules (contraction, interchange, and weakening).

This point is perhaps best conveyed in an intuitive, nontechnical way as follows: in TLG, the syntax has to keep track of word order and word occurrences, hence the need for a directional and linear syntactic type theory. But in HOG, the syntax is freed of this responsibility, because every time a syntactic word is used in a derivation, an occurrence of its phonological interpretation shows up, appropriately linearized, in the string of phonological words. Adapting the sort of economic metaphor favored by linear logicians: logical constants come for free, but every time you use a nonlogical constant in a syntactic proof, you have to pay for it with a spoken word (by saying its name out loud).

As is well known (Zaenen and Karttunen, 1984, Sag et al., 1985, Pullum and Zwicky, 1986) any syntactic theory must distinguish between ambiguity (two or more signs with the same phonology) and something else variously known as neutrality, nondistinctiveness, syncretism, indeterminacy, or underspecification. Here we sketch the HOG treatment of this distinction, starting with ambiguity. In the simplest case (so-called argument ambiguity), we have two distinct words with the same phonological interpretation, e.g., *bank* 'riverside' and *bank* 'financial institution':

$\mathsf{bank}_1 \in \mathrm{N}$
$\mathsf{bank}_2 \in \mathrm{N}$

General properties of the cartesian product (and the associated projection terms and pairing term constructor) ensure that the presence of these two lexical entries is equivalent to the presence of the single conjunctive specification

$(\mathsf{bank}_1, \mathsf{bank}_2) \in \mathrm{N} \times \mathrm{N}$

Of more interest is so-called functor ambiguity, where the two signs in question both have implicative types with the same consequent, e.g. main verb *can* and modal *can*:

a. I can tuna.
b. I can get a better job if I want to.
c.*I can tuna and get a better job if I want to.

In this case the pair of ambiguous words has type (ignoring morphosyntactic features)

$(\mathrm{NP} \Rightarrow \mathrm{VP}) \times (\mathrm{VP} \Rightarrow \mathrm{VP})$

which, by the intuitionistically valid law of disjunctive syllogism and its converse, is equivalent (in the sense that the functions denoted by the proofs in both directions are each other's inverses) to the type

$(\mathrm{NP} + \mathrm{VP}) \Rightarrow \mathrm{VP}$

Intuitively, since + (cartesian coproduct) is disjunction in the type logic, this says that *can* can take as its complement something which is either an NP or a VP (but not, as will be shown below, a coordination of an NP and a VP, which is neither). Formally, this treatment parallels the standard TLG treatment of ambiguity in Morrill (1990), which also employs cartesian coproduct (written $\vee$ in the Lambek calculus setting, where in terms of the type logic it is linear additive disjunction, in spite of being incorrectly characterized as boolean in most of the relevant TLG literature). Unfortunately, the standard TLG treatment of coordination (Morrill, 1990, Bayer and Johnson, 1995, Bayer, 1996, hereafter MBJ), which builds on Steedman's polymorphic typing of coordinate conjunctions as $A\backslash A/A$, wrongly generates such ungrammatical examples side by side with grammatical examples such as

John is rich and an excellent cook.

because coordinate structures are analyzed as having disjunctive types (here $\mathrm{AP} \vee \mathrm{NP}$, with the neutral functor *is* receiving the lexical type assignment $\mathrm{VP}/(\mathrm{AP} \vee \mathrm{NP})$). Thus the TLG account fails to distinguish functor ambiguity from functor neutrality. To take another example, the MBJ account predicts both of the following to be grammatical:

a. *Mary wants to go and John to go.

b. I would like to leave town early and for you to go with me.

As pointed out by Whitman (2002), the MBJ account also fails to distinguish between argument ambiguity and argument neutrality (which includes case syncretism as a special case), so that all instances of homophony between slots in the inflectional paradigm of a word are wrongly predicted to syncretize (see Dyla (1984) for relevant counterexamples). Moreover, the MBJ account is inconsistent with the standard TLG frame-semantics approach to phonological interpretation (Heylen, 1996, 1997, Moortgat, 1997, Carpenter, 1998). On that account, if S is the stringset that phonologically interprets AP and T is the stringset that phonologically interprets NP, then *rich and an excellent cook* should be in their union; hence it must lie in either S or in T; but

it does not. Faced with these difficulties, Whitman suggests abandoning the syntactic distinction between ambiguity and neutrality (so that in principle neutralization is always an option, subject only to pragmatic factors). Alternatively, Morrill (p.c.) suggests the possibility of distinct phonological entities with no audible difference (more precisely, a phonological entity is not just a string, but rather an ordered pair of a string and an integer).

Some of the problems discussed above have also been addressed within recent HPSG literature, most recently by Sag (2003), which proposes a relaxation of the requirement that feature structures be sort-resolved. In the absence of a precise formalization, this proposal is hard to assess. (Note that the model theory of RSRL precludes any entities which belong to a sort without belonging to one of its maximally specific subsorts.)

Pollard and Hana (2003) propose the following HOG analysis of neutrality and coordination of unlikes. First, the treatment of syncretism (say, for a language with nominative and accusative case) follows Levine et al. (2001) in employing a nonstandard inventory of Case values: pnom (pure nominative), pacc (pure accusative), and nom_acc (syncretic between nominative and accusative). Then NPnom and NPacc are defined as subtypes of NP as follows:

$$\text{NPnom} =_{def} [x \in \text{NP} \mid \textsc{Case}(x) = \textsf{pnom} \vee \textsc{Case}(x) = \textsf{nom\_acc}]$$
$$\text{NPacc} =_{def} [x \in \text{NP} \mid \textsc{Case}(x) = \textsf{pacc} \vee \textsc{Case}(x) = \textsf{nom\_acc}]$$

Also we define

$$\text{NPnom\_acc} =_{def} [x \in \text{N} \mid \textsc{Case}(x) = \textsf{nom\_acc}] = \text{NPnom} \cap \text{NPacc}$$

That is, case syncretism is handled not by the type logic's conjunction (cartesian product, which is appropriate only for non-neutralizing ambiguity) but rather by the (genuinely boolean) intersection of separation subtypes.

Pollard and Hana's analysis of coordination employs a schematic polymorphic type GEN[A] where A can be instantiated as any type of kind Sign. That is, for each sign type A, there is a type GEN[A] of "generalized A", where a generalized A is a sign that is either an A or a coordinate structure whose conjuncts are generalized $A's$. To ensure that, for each sign type A, A is actually a subtype of GEN[A], we add to the term logic a type-schematized family of constants $gen_A \in (A \Rightarrow \text{GEN}[A])$ together with a type-schematized set of constraints which ensure that in any model, each sign type is embedded in a one-to-one fashion into its generalization. (In the model, each of these constants is interpreted as a function that maps each sign of a certain type into a string of signs of length one). All that remains to complete the analysis

of coordination is to add type-schematized conjunctions (e.g., and_A and or_A) to the lexicon. What drives the analysis is the polymorphic typing of these constants, which is not $A \Rightarrow (A \Rightarrow A)$ (as would be suggested by the Steedman typing), but rather the type

$$\mathrm{GEN}[A]^+ \Rightarrow (\mathrm{GEN}[A] \Rightarrow \mathrm{GEN}[A]).$$

The phonological interpretation functor will ensure that the nonempty list argument shows up to the left of the conjunction and the other argument to the right. (Note that, in order for coordinate structures to serve as arguments to other signs, we must systematically retype our unsaturated lexical entries from $A \Rightarrow B$ to $\mathrm{GEN}[A] \Rightarrow B$, e.g. VP is redefined from NP $\Rightarrow$ S to GEN[NP] $\Rightarrow$ S.)

References

Barr, M. and C. Wells. 1999. *Category Theory for Computing Science 3rd edition*. Montreal: CRM.

Bayer, S. 1996. The coordination of unlike categories. *Language* 72(3):579–616.

Bayer, S. and M. Johnson. 1995. Features and agreement. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, vol. 33, pages 70–76. ACL.

Carpenter, B. 1998. *Type-Logical Semantics*. Cambridge, MA: MIT Press.

Carpenter, R. 1992. *The Logic of Typed Feature Structures*. New York: Cambridge University Press.

Church, Alonzo. 1940. A formulation of the simple theory of types. *Journal of Symbolic Logic* 5:56–68.

Curry, H. 1961. Some logical aspects of grammatical structure. pages 56–68.

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

Dyla, S. 1984. Across-the-board dependencies and case in Polish. *Linguistic Inquiry* 15(4):701–705.

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

Henkin, L. 1950. Completeness in the theory of types. *Journal of Symbolic Logic* 15:81–91.

Heylen, D. 1996. On the proper use of booleans in categorial logic. In *Proceedings of the Conference on Formal Grammar 1996*. Prague.

Heylen, D. 1997. Generalization and coordination in categorial grammar. In *Proceedings of the Conference on Formal Grammar 1997*. Aix-en-Provence.

Howard, W. 1980. The formulae-as-types notion of construction. pages 479–490.

Kepser, S. 2001. On the complexity of RSRL. In *Proceedings of FG-MOL 2001, Electronic Notes in Theoretical Computer Science 53*. Kluwer.

Lambek, J. 1958. The mathematics of sentence structure. *American Mathematical Monthly* 65:154–169.

Lambek, J. 1961. On the calculus of syntactic types. pages 166–178.

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

Lambek, J. 1999. Deductive systems and categories in linguistics. In H. J. Ohlbach and U. Reyle, eds., *Logic, Language, and Reasoning: Essays in Honour of Dov Gabbay*, pages 279–294. Dordrecht: Kluwer.

Lambek, J. and P. Scott. 1986. *Introduction to Higher Order Categorical Logic*. Cambridge: Cambridge University Press.

Levine, R., T. Hukari, and M. Calcagno. 2001. Parasitic gaps in English: some overlooked cases and their theoretical implications. pages 181–222. Cambridge, MA: MIT Press.

Montague, R. 1974. The proper treatment of quantification in ordinary English. In R. Thomason, ed., *Formal Philosophy*, pages 247–270. New Haven, CT: Yale University Press.

Moortgat, G. 1997. Categorical type logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*. New York: Elsevier.

Morrill, G. 1990. Grammar and logical types. In M. Stokhof and L. Torenvliet, eds., *Proceedings of the Seventh Amsterdam Colloquium*, pages 429–450. Amsterdam: Institute for Logic, Language, and Information.

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

Moshier, M.A. 1999. HPSG as type theory. In J. Ginzburg, L. Moss, and M. de Rijke, eds., *Logic, Language, and Computation*, vol. 2. Stanford, CA: CSLI.

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

Penn, G. and K. Hoetmer. 2003. *In search of epistemic primitives in the English Resource Grammar (or Why HPSG can't live without higher-order datatypes)*. East Lansing.

Pollard, Carl. 2004. Higher-order categorical grammar. In *Submitted for Categorial Grammar 2004*.

Pollard, Carl. in preparation. Hyperintensions in higher-order categorical logic Submitted for LoLa 2004.

Pollard, Carl and Jiri Hana. 2003. Ambiguity, neutrality, and coordination in higher-order grammar. In *Proceedings of Formal Grammar 2003*.

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

Pullum, G. and B. Scholz. 2001. On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In P. de Groote, G. Morrill, and C. Retoré, eds., *LACL 2001, LNAI 2099*, pages 17–43. Berlin: Springer-Verlag.

Pullum, G. and A. Zwicky. 1986. Phonological resolution of syntactic feature conflict. *Language* 62(4):751–773.

Ranta, A. In press. Grammatical Framework: a type-theoretical grammar formalism. *Journal of Functional Programming* .

Richter, F. 2000. *A Mathematical Formalism for Linguistic Theories with Application to Head-Driven Phrase Structure Grammar and a Fragment of German.*. Ph.D. thesis, University of Tübingen.

Richter, F. and M. Sailer. 2003. Basic concepts of lexical resource semantics. Course materials.

Sag, I. 2003. Coordination and underspecification. In J.-B. Kim and S. Wechsler, eds., *The Proceedings of the 9th International Conference on HPSG*, pages 267–291. Stanford: CSLI.

Sag, I., G. Gazdar, T. Wasow, and S. Weisler. 1985. Coordination and how to distinguish categories. *Natural Language and Linguistic Theory* 3:117–171.

Wheeler, D. 1981. *Aspects of a Categorial Theory of Phonology*. Ph.D. thesis, University of Massachusetts.

Whitman, P. 2002. *Category Neutrality: A Type-Logical Investigation*. Ph.D. thesis, Department of Linguistics, The Ohio State University.

Zaenen, A. and L. Karttunen. 1984. Morphological non-distinctiveness and coordination. In *Proceedings of the First Eastern States Conference on Linguistics*, pages 309–320.