Type-inheritance Combinatory Categorial Grammar

John Beavers
CSLI, Stanford University
Stanford, CA, 94305
jbeavers@csli.stanford.edu

Abstract

In this paper I outline Type-inheritance Combinatory Categorial Grammar (TCCG), an implemented feature structure based CCG fragment of English. TCCG combines the fully lexical nature of CCG with the type-inheritance hierarchies and complex feature structures of Head-driven Phrase Structure Grammars (HPSG). The result is a CCG/HPSG hybrid that combines linguistic generalizations previously only statable in one theory or the other, even extending the set of statable generalizations to those not easily captured by either theory.

1 Introduction

Type-inheritance Combinatory Categorial Grammar (TCCG) is a type-inheritance, unification-based CCG of the English fragment in Sag and Wasow (1999), implemented in the LKB (Copestake, 2002), a grammar development platform for producing efficient grammars for deep parsing. Typically, work in CCG (Steedman, 1996, 2000, *inter alia*) has focused on theoretical issues such as constituency and a principled syntax/semantics interface, with less work spent on the organization of grammatical information (see Baldridge 2002). Work in HPSG (Pollard and Sag, 1994; Ginzburg and Sag, 2000, *inter alia*) has instead focused on well-grounded structuring of grammatical information, most specifically in terms of type-inheritance hierarchies, although often at the cost of coverage and elegance (see Penn and Hoetmer 2003). However, the respective foci of work in these frameworks are largely orthogonal, suggesting a great potential in combining them, as recent work by Villavicencio (2001) and Baldridge (2002) has demonstrated. Following directly on this work, I adopt the type hierarchies of HPSG wholesale into TCCG, exploring directly the theoretical advantages this hybrid approach yields, with direct comparison to previous CCG and HPSG work. A full description of TCCG is beyond the scope of this paper (see Beavers 2002); I sketch below just a few of its advantages. In §2 I discuss background information about encoding TCCG in the LKB, including how TCCG deals with the so-called "spurious ambiguity" problem of CCG. In §3 I compare the relevant features of HPSG and CCG and discuss previous work that has combined them. In §4 I discuss the advantages type hierarchies bring to CCG, using the structure of the lexicon as well as the structure of lexical mapping rules as case studies. In §5 I discuss the advantages of CCG's highly lexicalist nature over common HPSG analyses and how these are encoded in TCCG. In §6 I discuss one domain (modification) where TCCG shows potential to simplify common analyses in both HPSG and CCG grammars due to its hybrid nature.

2 Implementation Details

I assume for this paper a rudimentary understanding of CCG. TCCG encodes as usual a small set of simplex syntactic categories (S, N, NP, PP, and CONJ) from which complex categories are built via slash operators. For example, *eat* is assigned category (S\NP)/NP, i.e. *eat* is a function from an NP to its right to a function from an NP to its left to S. The basic rule-set is outlined in (1):[1]

(1) (a) Forward Functional Application (>) :
$X/Y \quad Y \Rightarrow X$

(b) Backward Functional Application (<) :
$Y \quad X\backslash Y \Rightarrow X$

(c) Forward Functional Composition ($>B^n$) :
$X/Y \quad Y/Z_1.../Z_n \Rightarrow X/Z_1.../Z_n$ (for $n \leq 2$)

(d) Backward Functional Composition (<B) :
$Y\backslash Z \quad X\backslash Y \Rightarrow X\backslash Z$

(e) Forward Type Raising (>T) :
$NP_{nom} \Rightarrow S/(S\backslash NP_{nom})$

(f) Backward Type Raising (<T) :
$X \Rightarrow T\backslash(T/X)$

(g) Backward Crossed Substitution ($<S_x$) :
$Y/Z \quad (X\backslash Y)/Z \Rightarrow X/Z$ (where $X=S\backslash\$$)

(h) Coordination (Φ) :
$X \quad CONJ \quad X \Rightarrow X$

[1] The \$-notation indicates meta-categories: \$ stands for any list of categories (e.g. X\$ could be X\Y, X/Y, (X\Y)/Z, etc.), \\$ for a list of backward-slashed categories, and /\$ for a list of forward-slashed categories. Subscripts indicate category identity, e.g. $\$_1$ refers to the same list in all its uses in one category.

Note that >B is generalized to allow for composition of n-ary functions (but currently only for $n \leq 2$), and >T is restricted to nominative subject NPs (the only place in English where it is important). Turning to encoding, I assume a sign-based packaging of syntactic and semantic information:[2]

(2) (a)

$$\begin{bmatrix}\begin{array}{l} \textit{sign} \\ \text{ORTH } \textit{*diff-list-of-strings*} \\ \text{NF } \textit{nf} \\ \text{SS } \begin{bmatrix}\begin{array}{l} \textit{synsem} \\ \text{ROOT } \begin{bmatrix}\begin{array}{l}\textit{root-struct}\\ \text{RCAT } \textit{basic-category} \\ \text{FEATS } \textit{features}\end{array}\end{bmatrix} \\ \text{CAT } \textit{category} \\ \text{SEM } \textit{sem-struct}\end{array}\end{bmatrix} \\ \text{DTRS } \textit{list-of-signs}\end{array}\end{bmatrix}$$

(b)

Feature	Description
ORTH	Orthography
NF	Normal form feature
SS	Syntactic/Semantic info
ROOT	Root category info
RCAT	Simple root category
FEATS	Morphosyntactic features
CAT	Category information
SEM	Semantic information
DTRS	Phrasal daughters

Following Baldridge (2002), the root category is the final result of a category after all applications (e.g. S for a transitive verb (S\NP)/NP) and defines the morphosyntactic features of a category. Ignoring the details of the category type hierarchy, simplex categories are atomic types and complex categories are feature structures with a simplex result and a list of arguments as illustrated in (3).

(3) (a) Argument:

$$\begin{bmatrix}\begin{array}{l}\textit{argument}\\ \text{ARG } \textit{synsem}\\ \text{SLASH } \textit{slash-struct}\end{array}\end{bmatrix}$$

(b) Complex Category:

$$\begin{bmatrix}\begin{array}{l}\textit{complex-category}\\ \text{RESULT } \textit{category}\\ \text{ACTIVE } \textit{list}\big(\textit{args}\big)\end{array}\end{bmatrix}$$

(c) Example Argument:

$$\begin{bmatrix}\begin{array}{l}\textit{argument}\\ \text{ARG } \mid \text{ CAT } \textit{np}\\ \text{SLASH } \textit{back-slash}\end{array}\end{bmatrix}$$

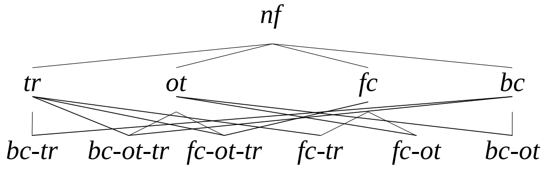
(Abbreviated as \NP)

(d) (((S\NP)/NP)/NP):

$$\begin{bmatrix}\begin{array}{l}\textit{complex-category}\\ \text{RESULT } \textit{s}\\ \text{ACTIVE } \big\langle \backslash\text{NP}, /\text{NP}, /\text{NP}\big\rangle\end{array}\end{bmatrix}$$

Finally, I briefly discuss how TCCG deals with the so-called “spurious ambiguity” of CCG. The combinatory power of CCG allows for a potentially exponential number of parses for a given reading of a single string.[3] A considerable amount of work has focused on spurious ambiguity and its effects on efficiency (see Karttunen 1986; see Vijay-Shankar and Weir 1990 for proof of a polynominal-time parsing algorithm and Clark and Curran 2004b for statistical models of CCG parsing), however most of these solutions are parser based. Rather than making proprietary modifications to the LKB's parser, I instead adopt Eisner's (1996) CCG normal form to eliminate spurious ambiguity. Eisner demonstrates that the parse forest assigned to a given string can be partitioned into semantic equivalence classes such that there is only one “canonical” (normal form) structure per equivalence class, where the normal form prefers application over B and right-branching >B over left-branching >B (and vice versa for <B).[4] These preferences are statable as constraints on what may serve as the primary functors of different combinators. I implement this by assigning one of the values in (4) to the feature NF:

(4)



An NF value *fc* marks a *sign* as being the output of >B, *bc* as the output of <B, *ot* as a lexical item or the output of application, and *tr* as the output of T. The subtypes are disjunctive, so that *fc-ot-tr* is either a lexeme or the output of >B, application, or T. Each combinator constrains the NF features of its output and daughters to be of specific value. For example, to prefer right-branching >B over left-branching >B, >B is constrained as in (5).

(5) $(\mathrm{X}/\mathrm{Y})_{bc-ot-tr} \quad \mathrm{Y}/\mathrm{Z} \Rightarrow \mathrm{X}/\mathrm{Z}_{fc}$

This constraint says that the output of >B is marked *fc* and its left daughter is *bc-ot-tr*, i.e. must be a lexical item or the output of <B, application, T, but not another >B (marked *fc*), thus ruling out left-branching >B over right-branching >B. Other combinators in (1) are constrained similarly. The cumulative effect results in only one “canonical” parse for each reading of a given string. For more discussion of the efficiency of this approach see Eisner (1996) and Clark and Curran (2004a). For purposes of TCCG, however, eliminating spurious ambiguity facilitates exploration of TCCG's hybrid nature by making direct comparisons possible between types of grammatical encoding in TCCG and more standard HPSG/CCG approaches, which I turn to next.

[2] In this paper I ignore the semantics of TCCG. It is worth noting that I do not adopt the λ-calculus semantics typical of CCG but opt instead for the Minimal Recursion Semantics (MRS) (Copestake et al., 1999) native to the LKB.

[3] However, the so-called “spurious” parses are in fact motivated by intonational and information structural phrases, as argued by Steedman (2000), although TCCG does not implement any prosody information.

[4] Eisner's constraints on $<\mathrm{S}_x$ only apply to CCGs with $<\mathrm{B}_n$ for $n > 2$ and are thus ignored. I do augment Eisner's system by restricting T to only occur when needed for B.

3 A Comparison of CCG and HPSG

In this section I briefly review some major differences between CCG and HPSG. Both theories share roots in the same strand of lexicalist syntax, wherein grammatical information is lexically encoded and combination is category driven. While the two theories differ considerably in several fundamental ways, there are two key differences relevant to this discussion. The first is how categories are constructed. In CCG the restricted set of simplex categories, the means by which complex categories are built, and the generality of the combinators collectively yield a principled system that conforms strongly to the lexicalist assumption that all combinatory information is encoded categorially. HPSG, however, allows a wide range of simplex categories and no restrictions on types of rules, allowing uneven divisions of combinatory information between categories and constructions. In principle a CCG style category/combinatory system is possible in HPSG (as TCCG demonstrates), but in practice large scale HPSGs tend to represent information heterogeneously, making certain cross-cutting generalizations difficult to state, largely a result of the directions HPSG has taken as a research program.

The second relevant difference between these theories is how categories are structured relative to one another. Traditionally, CCG offers no grammatical tools to statically relate categories. Instead, these relationships are left implicit even when linguistically relevant, only statable meta-theoretically. HPSG has from its inception employed multiple inheritance type hierarchies (e.g. as in (4)), where some of the grammatical information for a particular *sign* is inherited from its immediate supertype, which itself inherits grammatical information from which its supertype, and all types share inherited information with their sisters. The result is a richly structured set of relationships between linguistic units that reduces redundancy and can be exploited to state grammatical and typological generalizations.

As noted in §1, the respective advantages of these theories are compatible, and much previous work has exploited this fact. Use of unification (a core operation in HPSG) in CG dates at least as far back as Karttunen (1986, 1989), Uszkoreit (1986), and Zeevat (1988). Work on incorporating inheritance hierarchies into CCG is relatively more recent. Most notably Villavicencio (2001) implements a hybrid CCG/HPSG grammar in the LKB for purposes of exploring a principles and parameters acquisition model, defining parameters in terms of underspecified type hierarchies that the learner makes more precise during the learning process.[5] Moving beyond acquisition, Baldridge (2002) argues more generally for a type-hierarchy approach to the structure of a CCG lexicon so as to reduce redundancy and capture broader typological generalizations, although he does not explicitly flesh out this proposal.[6] With TCCG I build directly on this previous work by applying Villavicencio's type inheritance techniques to the issues raised by Baldridge, addressing head on the advantages of a hybrid approach and comparing it to prior HPSG and CCG analyses. In the following sections I outline several case studies of this approach.[7]

4 Advantages of TCCG over CCG

I turn first to the use of type hierarchies and lexical mapping rules in TCCG and the elimination of redundancy this brings to CCG. Using as my case study the hierarchy of verbal signs, in CCG the following categories are assigned to various verb types (note that in TCCG CPs are categorially finite NPs):

(6) (a) Intransitive (*sleep*): $\mathrm{S}\backslash\mathrm{NP}$

(b) Intransitive PP complement (*speak (to)*): $(\mathrm{S}\backslash\mathrm{NP})/\mathrm{PP}$

(c) Intransitive CP complement (*think*): $(\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP}_{fin}$

(d) Intransitive PP-CP complement (*say (to)*): $((\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP}_{fin})/\mathrm{PP}$

(e) Intransitive CP-subject (*suck*): $\mathrm{S}\backslash\mathrm{NP}_{fin}$

(f) Transitive verbs (*see*): $(\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP}$

(g) Transitive PP complement (*donate*): $((\mathrm{S}\backslash\mathrm{NP})/\mathrm{PP})/\mathrm{NP}$

(h) Transitive CP complement (*tell*): $((\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP}_{fin})/\mathrm{NP}$

(i) Ditransitive (*give*): $((\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP})/\mathrm{NP}$

(j) Subject control (*want/appear*): $(\mathrm{S}\backslash\mathrm{NP})/(\mathrm{S}\backslash\mathrm{NP})$

(k) Object control (*persuade/ask*): $((\mathrm{S}\backslash\mathrm{NP})/(\mathrm{S}\backslash\mathrm{NP}))/\mathrm{NP}$

(l) Auxiliary (*will*): $(\mathrm{S}\backslash\mathrm{NP})/(\mathrm{S}\backslash\mathrm{NP})$

[5] Note that TCCG employs a different type of CG than Villavicencio's implementation, which has generalized weak permutation and product categories but no type-raising.

[6] See also Erkan (2003) for a recent attempt to describe morphosyntactic features in CCG via type hierarchies.

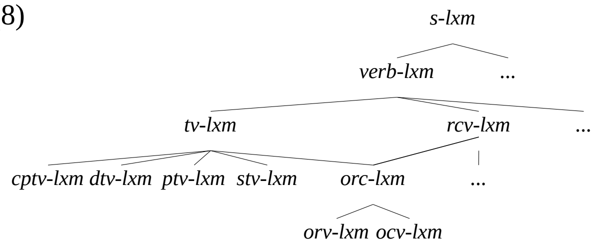
[7] Before proceeding I should note that TCCG is based primarily on Sag and Wasow (1999). This source was chosen for two reasons: (a) TCCG is primarily a proof-of-concept and thus a relatively constrained textbook grammar is ideally suited to exploring the issues addressed here and (b) a parallel HPSG implementation already exists that could provide for direct comparisons (although this is a matter of future work). However, development of TCCG has been informed by a wider range of work in CCG and HPSG and the conclusions I draw are applicable to both theories at large.

Of course, several linguistically relevant relationships hold across these types, as shown in (7).

(7) (a) All verbs share morphosyntactic features.
(b) All verbs have a leftward subject.
(c) All verbs obey obliqueness hierarchies (NPs are closest to verbs, obliques further, modulo syntactic operations like heavy-NP shift).
(d) All complements are rightward.
(e) Barring morphosyntax, auxiliary and control verbs share a category.

While these generalizations are of course derivable meta-theoretically (from the categories in (6), there is no explicit mechanism in CCG for stating static relationships (there are mechanisms for deriving categories, which I discuss below). TCCG, however, captures (7) via a lexical type hierarchy, the subtype for transitive verbs given in (8).[8]

(8)



Each *sign* in TCCG is assigned a type in such a hierarchy, where relevant generalizations in supertypes are inherited by subtypes. For example, the constraint that all verbs are rooted in S is stated on *s-lxm*, while the constraint that they all have leftward subjects is stated on *verb-lxm*:

(9) $$\textit{verb-lxm} := \begin{bmatrix}\text{SS} \mid \text{CAT} \mid \text{ACTIVE}\left\langle \backslash\text{NP}, \ldots\right\rangle\end{bmatrix}$$

Further specializations add additional information, for example *tv-lxm* adds information that there is at least one additional item in the valence of the verb $((S\backslash NP)/X\$)$. This type hierarchy has several advantages. First, it significantly reduces redundancy, since each constraint relevant for multiple categories is (ideally) stated only once. Second, these types provide a locus for cross-linguistic typological generalizations, an advantage that goes beyond parsimony. For example, the slash-marking constraint on *verb-lxm* in (9) defines English as an SV language. For a language like Irish this type could encode a general VS constraint (e.g. *verb-lxm* := $S/NP\$$). Thus the type hierarchy provides an explicit means for encoding broad typological parameters not directly statable in CCG (see Bender et al. 2002 for further discussion and Villavicencio 2001 on acquisition of word order parameters).

However, even (6) is not exhaustive of all possible verbal categories, since each verb carries not just its "basic" category but also a cluster of other categories corresponding to various lexical operations. For example, *give* is associated with several categories, including but not limited to:

(10) (a) Double object: $((S\backslash NP)/NP)/NP$
(b) NP-PP complement: $((S\backslash NP)/PP_{to})/NP$
(c) Passivized double object, no agent: $(S\backslash NP)/NP$
(d) Passivized double object with agent: $((S\backslash NP)/PP_{by})/NP$
(e) Passivized NP-PP complement, no agent: $(S\backslash NP)/PP_{to}$
(f) Passivized NP-PP complement with agent: $((S\backslash NP)/PP_{by})/PP_{to}$

Many standard CCG approaches encode these categories redundantly, although frequently these relationships are described via meta-rules (for instance as proposed by Carpenter 1992 and assumed implicitly in Steedman 2000). For instance, the meta-rule for dative shift could be stated as (11):

(11) $$((S\backslash NP_i)/NP_j)/NP_k \Rightarrow ((S\backslash NP_i)/PP_{to,k})/NP_j$$

This meta-rule simply says that any double-object verb will also have a dative-shifted category as well. The meta-rule approach is of course similar to the lexical mapping rules common in much HPSG literature (cf. Flickinger 1987, *inter alia*), and in fact the rule in (11) is implemented as in (12).

(12) $$\textit{dative-shift} := \begin{bmatrix} \text{SS} \mid \text{CAT} \begin{bmatrix} \text{RESULT } \boxed{1} \\ \text{ACTIVE} \left\langle \boxed{2}\backslash\text{NP}, /\text{PP}\begin{bmatrix}\text{AGR } \boxed{4}\\ \text{FORM } f\_to \\ \text{SEM } \boxed{5}\end{bmatrix}, \boxed{3}/\text{NP} \right\rangle \end{bmatrix} \\ \text{DTRS} \left\langle \begin{bmatrix} \text{SS} \mid \text{CAT} \begin{bmatrix}\text{RESULT } \boxed{1} \\ \text{ACTIVE} \left\langle \boxed{2}, \boxed{3}, /\text{NP}\begin{bmatrix}\text{SEM } \boxed{5}\\ \text{AGR } \boxed{4}\end{bmatrix}\right\rangle\end{bmatrix}\end{bmatrix} \right\rangle \end{bmatrix}$$

However, the difference between meta-rules and lexical rules is that the latter are first-class grammatical entities and can themselves can be organized

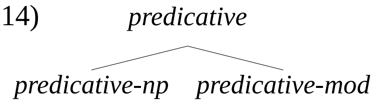
[8] I use the following type abbreviations: *s-lxm*=lexeme rooted in S, *n-lxm*=lexeme rooted in N, *verb-lxm*=verb, *tv*=transitive verb, *rcv*=control verb, *cptv*=CP complement transitive verb, *dtv*=ditransitive verb, *ptv*=PP complement transitive verb, *stv*=strictly transitive verb, *orc*=object control verb, *orv*=object raising verb, *ocv*=object equi verb.

hierarchically in a way that eliminates redundancy and captures several linguistic generalizations. An illustrative example is the encoding of predicative XPs (*Kim is happy/on time/the person who came*). TCCG adopts the Pollard and Sag (1994) analysis that predicative (ad)nominals have the category (S\NP) and thus are compatible with the selectional restrictions of *be* ((S\NP)/(S\NP)). A simple solution for generating predicative XPs is to derive $(S\backslash NP)\$_1$ categories from $NP\$_1$ (Det/lexical NPs) and $N|N\$_1$ (for Adjs/Ps) via the lexical rules in (13).

(13) (a) Predicative NPs: $NP_i\$_1 \Rightarrow (S\backslash NP_i)\$_1$
(b) Predicative adnominals: $N_i\ |N_i\$_1 \Rightarrow (S\backslash NP_i)\$_1$

These two rules clearly share a number of similarities that positing the two rules independently do not capture. In TCCG, however, the type hierarchy captures the larger similarities, where the rules for predicative NPs and predicative modifiers share a supertype that captures common information:

(14)



The type *predicative* encodes the general $\mathrm{Nom}\$ \Rightarrow (S_{pred+}\backslash NP)\$\ (S_{pred+}\backslash NP)\$$ form of the rules; *predicative-np* and *predicative-mod* merely further specify the daughter category as in (13). Again, while many CCG approaches employ meta-rules, the type hierarchy of TCCG allows further generalizations even among such meta-rules. In sum, the use of type hierarchies and lexical rules results in a grammar where each lexical item has (ideally) one category, with shared information stated once. Additional categories are derived via mapping rules, themselves organized hierarchically, thus capturing a variety of cross-cutting generalizations.

5 Advantages of TCCG over HPSG

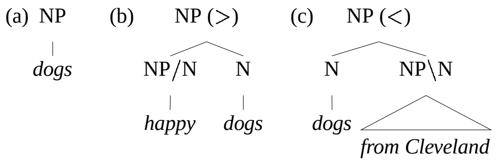
TCCG of course adopts wholesale the type-inheritance, unification based approach of HPSG, adding nothing new to the underlying framework. Nonetheless, by adopting a CCG style syntax TCCG makes possible more direct comparisons of the coverage and heavily lexical nature of standard CCG analyses to common HPSG approaches. Expanding the coverage over Sag and Wasow (1999), TCCG implements CCG analyses of a wide range of unbounded dependency phenomena (e.g. pied-piping, relative clauses, p-gaps, **that-t* effects; see Sag 1997, Ginzburg and Sag 2000 for well worked out HPSG analyses). More generally, TCCG implements CCG analyses of non-constituent coordination (e.g. right node raising and argument cluster coordination), largely unanalyzed in HPSG (although see Yatabe 2002, Chrysmann 2003, Beavers and Sag to appear). These are all well-known advantage of CCG and I will not discuss them at length.

In this section, however, I focus on how the fully lexical nature of TCCG simplifies the analysis of bare nominals, which in Ginzburg and Sag (2000) are analyzed constructionally: a plural/mass $\bar{N}$ is pumped to an NP with appropriate semantics (although see Beavers 2003 for an alternative HPSG proposal without pumping). The motivation for a phrasal pumping rule is to ensure (a) that modifiers may modify the N before the category is changed to NP and (b) that the added existential/generic quantifier outscopes all constituents of the N. For instance, to build the NP *happy dogs from Cleveland* in HPSG lexically would generate a lexical NP *dogs* incompatible with the constraints on modifiers like *happy* (which have $\bar{N}$ MOD values) and further would prevent the added quantifier to outscope the modifiers. However, a phrasal approach misses the broader generalization that these constructions are lexically triggered (by particular noun classes/inflection) and again heterogeneously spreads out language particular grammatical information between the lexicon and phrasal rules. At least in terms of parsimony a lexical rule approach would be preferred as it localizes the operation to one component of the grammar. CCG allows for such a fully lexical analysis of bare plurals. The relevant categories are shown in (15):

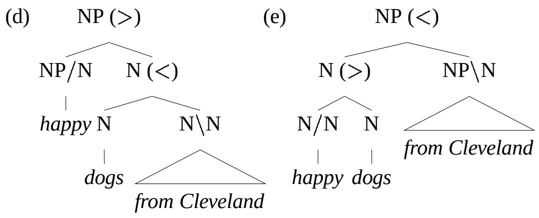
(15) (a) Nouns: $N\$$
(b) Attributive adjectives: $N/N\$$
(c) Attributive prepositions: $N\backslash N/NP$
(d) Relativizers: $(((N\backslash N)/\$_1)/(S/\$_1/NP))\$_2$
(d) Determiners: NP/N

N, Adj, Rel, and P are all of form N\$, with only Det rooted in NP. Adopting Carpenter's (1992) meta-rule analysis of bare NPs to TCCG, I analyze bare nominals via a simple HPSG-style lexical rule of the form in $N\$_1 \Rightarrow NP\$_1$ such that (ad)nominal can be pumped to a function rooted in NP (adding the appropriate quantificational semantics), essentially making them determiners. Thus when building a bare NP the pumped category is necessarily the final functor, ensuring no category mismatches and the correct semantics, as shown in (16).[9]

(16)



[9] I represent derivations as trees rather than the usual CCG notation to be consistent with the LKB's tree-based output. Note that the normal form described in §2 rules out (16e).



A variety of other phenomena have been implemented lexically in TCCG without the use of additional syntactic rules above and beyond the ones assumed above in §2, reducing the number of different kinds of syntactic and constructional rules common in HPSG analyses. Thus, TCCG validates and makes more accessible the possibilities of fully lexical CCG-style analyses in HPSG without modifying the underlying framework.

6 Advantages over both HPSG and CCG

One advantage over both HPSG and CCG comes in the treatment of modifiers. In most HPSG literature modifiers form a heterogeneous class: due to the unconstrained possibilities of category formation, the HEAD category and the *synsem* in MOD are not inherently related and thus do not necessarily allow for any further generalizations. In CCG, however, modifiers all have the general form X|X$, where X is typically a basic category (Adjs are of category N/N$, Ps are N\N$, Advs are S|S$ (ignoring VP-Advs)). Yet this generalization is not codifiable in CCG terms and each modifier must redundantly encode the same form. In TCCG, however, I posit a type *xp-mod-lxm* that characterizes these generalizations over modifiers of basic categories:

(17) X|X$, X a simplex category:
xp-mod-lxm :=

$$\begin{bmatrix} \text{SS} & \begin{bmatrix} \text{ROOT} \mid \text{FEATS}\ \boxed{1} & \\ \text{CAT} & \begin{bmatrix} \text{RESULT}\ \boxed{2} & \\ \text{ACTIVE} & \left\langle \begin{bmatrix} \text{ARG} & \begin{bmatrix} \text{ROOT} \mid \text{FEATS}\ \boxed{1} \\ \text{CAT}\ \boxed{2} \end{bmatrix} \end{bmatrix}, \ldots \right\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Here the category and morphosyntactic features of the first argument are shared with the result, with the rest of the arguments left underspecified, capturing the general nature of modifiers in TCCG.[10] The advantage to the type hierarchy here is that most of the relevant information about each kind of modifier is now only stated once. Subtypes of this type need only add relevant additional information, for instance the supertype of all adjectives, *adj-lxm*, inherits from both *xp-mod-lxm* (meaning it's a modifier) and *nom-lxm* (meaning it's rooted in N), adding only the constraint that slash in X|X$ be forward:

(18) Adjectives are N/N$:
adj-lxm := $\begin{bmatrix} \text{SS} \mid \text{CAT} \mid \text{ACTIVE} \left\langle / \, [\,], \ldots \right\rangle \end{bmatrix}$

Transitive and intransitive subtypes of *adj-lxm* further specialize the $, and similar structuring of information occurs for all other modifier types. Thus the commonalities and differences of a wide variety of modifiers are captured in terms of type hierarchies, potentially with typological advantages. In Romance languages such as Spanish, where adnominal modifiers are overwhelmingly post-head, the directionality constraint for adjectives in (18) could instead be stated as a default on a higher supertype of all adnominals (where the few exceptions lexically override the default). Again, these types of constraints are not possible in most HPSG or CCG implementations. CCG without type hierarchies lacks the language in which such generalizations can be stated. Instead modifiers only form a class meta-theoretically with shared information stated redundantly. On the other hand, most HPSG approaches typically do not offer a sufficiently constrained set of category types to state generalizations over modifiers. Generalizations over modifier classes must be stated heterogeneously as a combination of lexical marking and pre- and post-head adjunct constructions (or alternatively stated in terms of independent linear precedence rules (Kathol, 2000)). Thus combining these approaches yields potential not easily realizable separately.

7 Conclusion

TCCG is an implemented CCG in an HPSG framework that combines the advantages of both theories: well-organized, minimally redundant lexical and grammatical information mixed with the theoretical elegance of CCG grammars. The combination allows for simplifications of common analyses in both theories, even allowing for generalizations that are more difficult to state in both theory. The details discussed here are just a subset of TCCG; for a full description see Beavers (2002).

Acknowledgements

The bulk of this work was done at the University of Edinburgh in 2002, with the support of The Edinburgh-Stanford Link Grant #2KBA905 and the LinGO project at CSLI at Stanford University.

[10] This is a simplification of the approach actually implemented in TCCG, which enriches the slash values of all categories with modalities indicating the “semantic” headedness of the category, following Baldridge (2002) and Kruijff (2001), providing further generalizations over modifiers, but the details are irrelevant for this discussion.

I'd like to thank Jason Baldridge, Tim Baldwin, Colin Bannard, Chris Callison-Burch, Ann Copestake, Dan Flickinger, Julia Hockenmaier, Martin Kay, Geert-Jan Kruijff, Stephan Oepen, Ivan Sag, Mark Steedman, Maarika Traat, Aline Villavicencio, Michael White, and several anonymous reviewers for their comments. I'd like to especially acknowledge Ann Copestake and Aline Villavicencio's earlier CCG LKB implementation as an immediate predecessor and influence on this one even if the two diverge significantly. Any mistakes or omissions are purely my own.

References

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

John Beavers and Ivan A. Sag. To appear. Some arguments for coordinate ellipsis in HPSG. In *Proceedings of the 2004 HPSG Conference, Katholike Universiteit Lueven, Belgium*, Stanford, CA. CSLI Publications.

John Beavers. 2002. A CCG implementation for the LKB. LinGO Working Paper #2002-8, CSLI, Stanford University, Stanford, CA.

John Beavers. 2003. More heads and less categories: A new look at noun phrase structure. In *Proceedings of the 2003 HPSG Conference, East Lansing, MI*, Stanford, CA. CSLI Publications.

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precisions grammars. In John Carroll, Nelleke Oostdijk, and Richard Sutcliffe, editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

Bob Carpenter. 1992. Lexical and unary rules in categorial grammar. In Bob Levine, editor, *Formal Grammar: Theory and Implementation*. Oxford University Press.

Stephen Clark and James R. Curran. 2004a. The importance of supertagging for wide-coverage ccg parsing. In *Proceedings of COLING 2004*, Geneva.

Stephen Clark and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the ACL*, Barcelona.

Ann Copestake, Dan Flickinger, Ivan Sag, and Carl Pollard. 1999. Minimal recursion semantics: An introduction. http://www.cl.cam.ac.uk/~acc10/papers/newmrs.ps.

Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford, CA.

Berthold Crysmann. 2003. An asymmetric theory of periphal sharing in HPSG: Conjunction reduction and coordination of unlikes. In *Formal Grammar 2003*. http://cs.haifa.ac.il/~shuly/fg03/.

Jason Eisner. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th Annual Meeting of the ACL*, Santa Cruz, June.

Güneş Erkan. 2003. A type system for combinatory categorial grammar. Master's thesis, The Middle East Technical University.

Daniel Flickinger. 1987. *Lexical Rules in the Hierarchical Lexicon*. Ph.D. thesis, Stanford University.

Jonathan Ginzburg and Ivan A. Sag. 2000. *Interrogative Investigations: The Form, Meaning, and Use of English Interrogatives*. CSLI Publications.

Lauri Karttunen. 1986. D-PATR: A development environment for unificated-based grammars. Csli report, CSLI, Stanford.

Lauri Karttunen. 1989. Radical lexicalism. In Mark Baltin and Anthony Kroch, editors, *Alternative Conceptions of Phrase Structure*. University of Chicago Press, Chicago.

Andreas Kathol. 2000. *Linear Syntax*. Oxford University Press, Oxford.

Geert-Jan M. Kruijff. 2001. *A Categorial-Modal Architecture of Informativity: Dependency Grammar Logic and Information Structure*. Ph.D. thesis, Charles University, Prague.

Gerald Penn and Kenneth Hoetmer. 2003. In search of epistemic primitives in the english resource grammar (or why HPSG can't live without higher-order datatypes). In Stefan Müller, editor, *Proceedings of the Tenth International Conference on Head-Driven Phrase Structure Grammar*. CSLI Publications.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago, Chicago, IL.

Ivan A. Sag and Thomas Wasow. 1999. *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford, CA.

Ivan A. Sag. 1997. English relative clause constructions. *Journal of Linguistics*, (33).

Mark Steedman. 1996. *Surface Structure and Interpretation*. MIT Press, Cambridge, Mass.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, Mass.

Henk Uszkoreit. 1986. Categorial unification grammars. In *Proceedings of COLING 1986*, Bonn, Germany.

K. Vijay-Shanker and David Weir. 1990. Polynomial time parsing of combinatory categorial grammars. In *Proceedings of teh 28th Annual Meeting of the Association for Computational Linguistics, Pittsburgh*, pages 1–8, San Francisco, CA. Morgan Kaufmann.

Aline Villavicencio. 2001. The acquisition of a unification-based generalized categorial grammar. Technical report, Computer Laboratory, Cambridge University.

Shuichi Yatabe. 2002. A linearization-based theory of summative agreement in peripheral-node raising constructions. In *Proceedings of the 2002 HPSG Conference, Kyung Hee University, Seoul*, Stanford, CA. CSLI Publications.

Henk Zeevat. 1988. Combining categorial grammar and unification. In *Natural Language Parsing and Linguistic Theories*, pages 202–229. Reidel, Dordrecht.