Categorial Grammar, v2.5

Jason Baldridge and *Frederick Hoyt*

Note: This is a draft chapter for the *Handbook of Syntax*. Please cite as:

Baldridge, Jason and Frederick Hoyt. (to appear). *Categorial grammar*. In Kiss, Tibor and Alexiadou, Artemis (eds.). Handbook of Syntax. Berlin: de Gruyter.

The handbook is likely to be published in 2012. Email `jbaldrid@mail.utexas.edu` with comments, corrections, or suggestions for this chapter.

Categorial grammar is an umbrella term for a family of grammatical formalisms which handle syntactic and semantic analysis via type-dependent analysis. Syntactic types and semantic interpretations are assigned to complex expressions that are compositionally determined from the types and interpretations of their subexpressions. Modern categorial grammar was first proposed by Ajdukiewicz (1935) as his "calculus of syntactic connection." It arose from the theory of semantic categories developed by Edmund Husserl and the Polish school of logicians, which included Ajdukiewicz and Stanislaw Lesniewski (Casadio, 1988). Bar-Hillel (1953) provided an order-sensitive formulation of categorial grammar, and Lambek (1958) provided its first formulation as a logic. Since then, categorial grammar has been extensively developed into variants that have greater descriptive, explanatory and computational adequacy for dealing with natural language grammar. This chapter focuses on the two current main branches of categorial grammar: Combinatory Categorial Grammar (Ades and Steedman, 1982; Szabolcsi, 1992; Jacobson, 1992b; Steedman, 1996a, 2000b; Steedman and Baldridge, 2011; Steedman, 2012) and Categorial Type Logics (also known as Type Logical Grammars) (van Benthem, 1989; Morrill, 1994; Moortgat, 1997; Carpenter, 1998; Vermaat, 1999; Oehrle, 2011).

Compared with other grammar formalisms such as Head-Driven Phrase-Structure Grammar (HPSG: Pollard and Sag (1994); Sag *et al.* (2003)) or Lexical-Functional Grammar (LFG: (Kaplan and Bresnan, 1982)), categorial grammars are *extreme* lexicalist formalisms, meaning nearly all grammatical information is contained within the entries of the lexicon while syntactic derivation is modeled with a small set of very general rules. In this respect, categorial grammars share common ground for proposals within the Minimalist Programs (MP: Chomsky (1995)), according to which syntactic derivation involves a very small number of rules operating over lexical categories richly specified with syntactic information.

Another core property of CG frameworks is semantic transparency. Syntactic category types (such as `s\np`) correspond directly to semantic types, typically expressed as terms of the lambda calculus (such as $\lambda y.\lambda x.[Pxy]$). Meaning

assembly is therefore strictly compositional. Essentially, syntactic categories simply constrain the linear order in which semantic functions can combine with their arguments. Syntactic structure itself is simply an artifact of derivation, not a level of representation (Steedman and Baldridge, 2011).

Typically, CG frameworks allow for much freer surface constituency than is typically assumed in context-free grammar formalisms (as well as in Ajdukiewicz's original formalism). This provides straightforward analyses of many interesting problems in syntax, including unifying analyses of unbounded constructions such as coordination and relative clause formation, an analysis of intonation structure as part of surface syntactic derivation, and algorithms that allow for incremental processing as a part of syntactic derivation with the competence grammar.

This chapter presents the very earliest form of categorial grammar, the AB calculus (Ajdukiewicz, 1935; Bar-Hillel, 1953), and then discusses predominant modern categorial grammar frameworks in a way that emphasizes their similarities rather than their differences. The AB calculus provides an intuitive and particularly simple example of categorial grammar that allows many of the core intuitions and mechanisms of the approach to be explicated in terms that will be more familiar to newcomers than its modern descendants.

Throughout the discussion, examples are provided of some of the linguistic analyses which have been proposed for phenemona such as long-distance relativization, right-node raising, argument cluster coordination, and parasitic gaps. Finally, brief discussion is provided of generative capacity, the connections to other frameworks, and recent computational work based on categorial grammar.

1 The Ajdukiewicz-Bar-Hillel Calculus

The common starting points of categorial grammar formalisms are Ajdukiewicz's (1935) syntactic connection calculus and Bar-Hillel's (1953) directional adaptation thereof. It is thus generally referred to as the AB calculus. It has two components: categories and rules.

SYNTAX: Categories may be either atomic categories (such as s for sentences and n for nouns) or complex categories which are functions defined over atomic categories and/or other complex categories. Complex categories directly encode the subcategorization requirements of the expressions they are associated with, along with specifications of the linear order in which arguments will be found. The linear order of arguments is encoded by means of "slash" operators $\backslash$ and $/$, which mean "find an immediately preceding argument" and "find an immediately following argument" respectively.

A simple example is the transitive verb category $(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}$ for English verbs. In words, this category indicates that it seeks a noun phrase to its right (with rightward leaning slash "/" and category np), then another noun phrase to its left (via the leftward leaning slash "\" and category np); upon consuming both of these arguments, it yields a sentence (s). [1]

[1] Note that the categories provided above use the "rightmost" notation for categories com-

English verbal categories, which will be used in later derivations, include:

(1) intransitive verbs: s\np

(2) transitive verbs: (s\np)/np

(3) ditransitive verbs: ((s\np)/np)/np

(4) sentential complement verbs: (s\np)/s

Categories are combined by means of two directionally sensitive rules of function application, *forward application* and *backward application*:

(5) a. $X/Y : f \quad Y : x \quad \Rightarrow_{>} \quad X : f(x)$ (FA)

b. $Y : x \quad X\backslash Y : f \quad \Rightarrow_{<} \quad X : f(x)$ (BA)

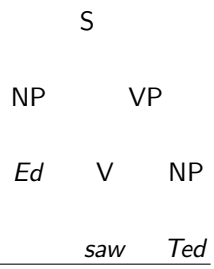
In words, forward application allows two word sequences, ω_a with category of type X/Y and interpretation f of type $\sigma\tau$ and ω_b with category of type Y and interpretation x of type σ, to form the word sequence $\omega_a + \omega_b$ of type X and interpretation $f(x)$ of type τ. Backward application is just the directional converse.

With the application rules, and a lexicon containing *Ed* and *Ted* with category np and *saw* with transitive category (s\np)/np, the derivation for *Ed saw Ted* is (6). Each step is annotated by underlining the combining categories and labeling the underline with the symbol for the rule used: ">" for forward application and "<" for backward application.

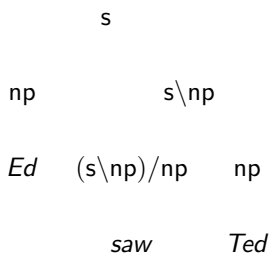
(6)
$$\dfrac{\dfrac{\textit{Ed}}{\mathsf{np}} \quad \dfrac{\dfrac{\textit{saw}}{(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}} \quad \dfrac{\textit{Ted}}{\mathsf{np}}}{\mathsf{s}\backslash\mathsf{np}}\,{>}}{\mathsf{s}}\,{<}$$

The application rules can be viewed as general binary phrase structure rules written in reverse (Steedman, 2000b). In this light, the above derivation can be seen as a phrase structure tree turned upside-down, isomorphic to the standard phrase structure analysis:

(7) a.



b.



monly employed in work in Combinatory Categorial Grammar. In this notation, the leftmost category in a function type is the result category: In (s\np)/np, s is the leftmost category in the function and it indicates the result of applying the verb's category to its two np arguments. An alternative notation used in the Lambek calculus tradition places leftward arguments to the left of the result category. As such, the transitive category is written (np\s)/np. There are advantages and disadvantages to both conventions which we will not address here, but the reader should expect to see both alternatives in the literature.

Despite this superficial similarity between derivations in the AB-calculus and context-free phrase-structure trees, the categories labeling the nodes of (7b) are more explicit than the atomic symbols of (7a). Subcategorization is directly encoded in functor categories rather than through the use of new symbols such as $\mathsf{V}_{intrans}$, V_{trans} and $\mathsf{V}_{ditrans}$ (although expressions like this are frequently used in the CCG literature as abbreviations for complex categories).

Furthermore, there is a systematic correspondence between notions such as *intransitive* and *transitive* — after the transitive category $(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}$ consumes its object argument, the resulting category $\mathsf{s}\backslash\mathsf{np}$ is that of an intransitive verb (see Oehrle (2011) for a deductive explanation of the relationship between categories and phrase structure labels).

FEATURES: Atomic categories need not be treated as simple labels, but can also be further elaborated with features, such as $\mathsf{s}[\mathsf{verb\_form}{=}\mathsf{finite}]$ and $\mathsf{np}[\mathsf{num} = \mathsf{plural}]$ to distinguish different subtypes of those categories. This allows agreement and other restrictions to be encoded in categories. For example, the category of *walks* would be $\mathsf{s}[\mathsf{verb\_form}{=}\mathsf{finite}]\backslash\mathsf{np}[\mathsf{num}{=}\mathsf{singular}, \mathsf{person}{=}\mathsf{3rd}]$ to ensure that it can only take a singular third-person subject noun phrase. In this article, agreement concerns of this nature are mostly ignored. When used, features are abbreviated as subscripts, e.g., $\mathsf{s}_{\mathsf{fin}}$ and $\mathsf{np}_{\mathsf{sing}}$. It should be noted that such features in CCG do not take on feature structures as values, as they do in HPSG and other unification based formalisms.

SEMANTICS: Categorial grammar frameworks assume a close connection between syntactic types and semantic types. In general, the syntactic type (category) of a word is determined by the semantic type of the predicate which the word realizes. For example, for a two-place predicate like **see** of type $e \rightarrow (e \rightarrow t)$ the corresponding transitive syntactic category is $(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}$. Given that the types of the atomic categories s, np, and n are t, e, and $e{\rightarrow}t$, respectively, and that $\backslash$ and / are directional variants of $\rightarrow$, the type mapping can be seen as:

(8) $e \rightarrow (e \rightarrow t) \Rightarrow \mathsf{np} \rightarrow (\mathsf{np} \rightarrow \mathsf{s}) \Rightarrow (\mathsf{np} \rightarrow \mathsf{s})/\mathsf{np} \Rightarrow (\mathsf{s}\backslash\mathsf{np})/\mathsf{np}$

The categories and the lambda expressions are both *curried* (*schönfinkelized*) functions which take their arguments one at a time (Schönfinkel, 1924). This pairs the λx and λy in $\lambda x \lambda y.\mathbf{see}(y, x)$ with the outermost np and innermost np, respectively, of the category $(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}$. The semantics given for these categories are of course vastly simplified and are intended only to indicate basic predicate-argument dependencies.

For verb-initial languages such as Modern Standard Arabic (9a) and verb-final languages such as Japanese (9b) the transitive categories would be $(\mathsf{s}/\mathsf{np})/\mathsf{np}$ and $(\mathsf{s}\backslash\mathsf{np})\backslash\mathsf{np}$, respectively:

(9) a. *ra'a* "(he) saw" $\vdash (\mathsf{s}/\mathsf{np})/\mathsf{np} : \lambda x.\lambda y.see'(x, y)$

b. *mimashita* "saw" $\vdash (\mathsf{s}\backslash\mathsf{np})\backslash\mathsf{np} : \lambda y.\lambda x.see'(x, y)$

A common choice for representing semantic expressions with categorial grammars is the lambda calculus (for alternatives, see Zeevat *et al.* (1987), Baldridge

and Kruijff (2002) and Copestake *et al.* (2001)). Some sample entries, in the format [*word* := category : **semantics**], are given below:

(10) a. *Ed* := np : **Ed**

b. *Ted* := np : **Ted**

c. *saw* := (s\np)/np : $\lambda x \lambda y.\mathbf{see}(y, x)$

The application rules allow a category like s\np to combine with an np argument to its left, reducing to s. Similarly, a lambda expression such as $\lambda x.\mathbf{walk}(x)$ applies to an argument like **Ed**, reducing to **walk**(**Ed**). In both of these dimensions, a functor of type $e \rightarrow t$ (from entities to truth values) applies to an argument of type e to produce a result of type t. This correspondence naturally suggests that the application rules should simultaneously reduce both categories and semantics in this manner.

With the entries in (10), a derivation with compositionally constructed logical forms for *Ed saw Ted* follows:

(11)
$$\dfrac{\dfrac{\textit{Ed}}{\mathsf{np} : \mathbf{Ed}} \quad \dfrac{\dfrac{\textit{saw}}{(\mathsf{s}\backslash\mathsf{np})/\mathsf{np} : \lambda x \lambda y.\mathbf{see}(y, x)} \quad \dfrac{\textit{Ted}}{\mathsf{np} : \mathbf{Ted}}}{\mathsf{s}\backslash\mathsf{np} : \lambda y.\mathbf{see}(y, \mathbf{Ted})}>}{\mathsf{s} : \mathbf{see}(\mathbf{Ed}, \mathbf{Ted})}<$$

Note that these lambda terms have been reduced implicitly in this derivation, e.g. $[\lambda x \lambda y.\mathbf{see}(y, x)]\mathbf{Ted} \rightarrow \lambda y.\mathbf{see}(y, \mathbf{Ted})$. Again, it should be stressed that these particular logical forms are meant to serve illustrative purposes, and are by no means a serious proposal for the meaning of such expressions.

Complex arguments: Categories may themselves have complex categories as their arguments. This is typically the case for categories of control verbs, relative pronouns and *wh*-items. Consider the following lexical entries (where *inf* and *base* indicate infinitival and base forms, respectively):

(12) *promised* := $((\mathsf{s}\backslash\mathsf{np})/(\mathsf{s}_{\mathsf{inf}}\backslash\mathsf{np}))/\mathsf{np} : \lambda x \lambda P \lambda y.\mathbf{promise}(y, x, Py)$

(13) *persuaded* := $((\mathsf{s}\backslash\mathsf{np})/(\mathsf{s}_{\mathsf{inf}}\backslash\mathsf{np}))/\mathsf{np} : \lambda x \lambda P \lambda y.\mathbf{persuade}(y, x, Px)$

(14) *to* := $(\mathsf{s}_{\mathsf{inf}}\backslash\mathsf{np})/(\mathsf{s}_{\mathsf{base}}\backslash\mathsf{np}) : \lambda P.P$

(15) *go* := $\mathsf{s}_{\mathsf{base}}\backslash\mathsf{np} : \lambda x.\mathbf{go}(x)$

Complex arguments such as $\mathsf{s}_{\mathsf{inf}}\backslash\mathsf{np}$ correspond to semantic predicates of type $e \rightarrow t$, indicated with uppercase variables (e.g. P above). The semantic expressions for subject or object control verbs reflects the application of that function to a variable representing either the subject or the object (above, y as subject versus x as object). This establishes co-indexation of the controlled subject with either the matrix subject or object.

For example, the derivations for *Ed promised Ted to go* and *Ed persuaded Ted to go* are syntactically identical, but the correct dependencies are established due to the co-indexation patterns given in the verb semantics.

(16)

Ed	*promised*	*Ted*	*to*	*go*
np	$((s\backslash np)/(s_{inf}\backslash np))/np$	np	$(s_{inf}\backslash np)/(s_{base}\backslash np)$	$s_{base}\backslash np$
		>	>	
	$(s\backslash np)/(s_{inf}\backslash np)$		$s_{inf}\backslash np$	
			>	
	$s\backslash np$			
			<	
s				

Subject relative clauses in English follow a similar pattern. For example, *who* has the category $(n\backslash n)/(s\backslash np)$, which takes the complex argument $s\backslash np$, after which it has the category of a post-nominal modifier $n\backslash n$. This can be seen in the following abbreviated derivation:

(17) man_{n} $\text{who}_{(n\backslash n)/(s\backslash np)}$ $[\text{saw Ted}]_{s\backslash np}$

Given the logical form $\lambda P \lambda Q \lambda x.(Px \wedge Qx)$ for *who*, the string *man who saw Ted* has the intersective interpretation $\lambda x.\mathbf{see}(x, \mathbf{Ted}) \wedge \mathbf{man}(x)$ (i.e., the entity x such that x is the subject of seeing Ted and x is a man).

Coordination: The type-driven nature of categorial grammar provides a natural explanation for basic coordination phenomena (see Hartmann, this volume). The intuition common to most grammatical frameworks is that like coordinates with like. This intuition can be represented as $(X\backslash X)/X$, where the variable X is used as a short-hand notation to schematize over the categories (giving us specific categories such as $(s\backslash s)/s$, $((n\backslash n)\backslash(n\backslash n))/(n\backslash n)$, and so on). This captures a number of basic coordination data points:

(18) Ed $[\text{saw Ted}]_{s\backslash np}$ and $[\text{walked}]_{s\backslash np}$.

(19) Ed $[\text{saw}]_{(s\backslash np)/np}$ and $[\text{heard}]_{(s\backslash np)/np}$ Ted.

(20) *Ed $[\text{saw Ted}]_{s\backslash np}$ and $[\text{heard}]_{(s\backslash np)/np}$.

(21) Ed $[\text{bought}]_{(s\backslash np)/np}$ and $[\text{gave Ted}]_{(s\backslash np)/np}$ books.

The fact that categories are typed, structured objects provides a direct account of constituent similarity. In context-free grammar, coordination is limited to constituents dominated by the atomic non-terminal labels defined in the grammar. For example, if ditransitives are captured with the rule $VP \rightarrow V\ NP\ NP$, then there is no simple constituent label for a string like *gave Ted*. A binary context-free grammar would remedy this particular example, but as is shown in following sections, categorial grammars will often posit much more radical alternative constituents.

Limitations of the AB calculus: The AB calculus has a number of appealing properties, but it is a non-associative system that has a number of limitations. For example, with the categories suggested so far, it cannot handle constructions which suggest a constituent analysis of the combination of subjects with verbs, such as:

(22) Right-node raising: (Ed saw) and (Ned heard), Ted.

(23) Object relative clauses: the man that (Ed saw).

(24) Intonation: Q: Who did Ed see? A: (Ed saw) TED.

If a grammar is to provide an interpretation for such meaningful substrings, then it would be natural to assume that a string such as *Ed saw* should have a derivation that delivers a logical form such as $\lambda x.\mathbf{see}(\mathbf{Ed}, x)$. The category then should be something seeking the object noun phrase to its right, but having already consumed its subject noun phrase—in other words, $\mathsf{s/np}$. This comes down to a question of *associativity*: just as $3 + (4 + 1) = (3 + 4) + 1 = 8$, can our grammar show that $\mathit{Ed} + (\mathit{saw} + \mathit{Ted}) = (\mathit{Ed} + \mathit{saw}) + \mathit{Ted} = \mathsf{s}$?

Another problematic set of constructions are those involving permutation, which is of course quite common in languages with scrambling, such as Turkish. In English, we see it with heavy-NP shift: *Ed saw yesterday his tall old friend Ted.* This relates to *commutativity* (e.g., $(3 + 4) = (4 + 3) = 7$).

The problem is that the AB calculus is too inflexible because it is both non-associative and non-commutative. While it is possible to deal with some of its limitations by increasing categorial ambiguity (e.g., using a further transitive category $(\mathsf{s/np})\backslash\mathsf{np}$ to handle (22)-(24)), it is worthwhile considering more powerful systems of categorial inference that can incorporate associativity and commutativity in a principled, rather than *ad hoc*, manner. In the next two sections, we consider two independently motivated ways to do so—rule-based and logical extensions—and also highlight their similarities.

2 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) developed from work by Lyons, Geach, Bach, Dowty and others in the 1960's and 1970's that extended the AB calculus with further rules (Steedman, 1996a, 2000b; Steedman and Baldridge, 2011; Steedman, 2012). There are at least two versions of CCG found in the literature. One is the formalism developed by Mark Steedman and his collaborators which incorporates type-raising, function composition and substitution rules (corresponding to Curry and Feys' T, B and S combinators). The other prominent approach to CCG has largely been developed in a series of papers by Pauline Jacobson (Jacobson, 1990, 1992a, 1993, 1999, 2000, 2003, a.o.) which makes extensive use of unary type-changing rules (such as the so-called Geach rule) in place of function composition. We begin by considering in turn the combinators incorporated into CCG by Steedman and his collaborators.

2.1 Combinatory rules

The rules that Steedman proposed for CCG are based on a small set of *combinators* from combinatory logic (Curry and Feys, 1958). Combinatory logic (Schönfinkel, 1924; Curry and Feys, 1958) is a formalism which allows functions defined in the lambda calculus to be manipulated without requiring the use of lambda-abstraction and hence of computation-intensive mechanisms for keeping track of variable assignments. Combinators are therefore functions defined over terms of the lambda-calculus.

The combinators used in CCG include: (function) composition (**B**), type-raising (**T**), and substitution (**S**). The relationships of these combinators to terms of the lambda calculus are defined by the following equivalences (Steedman, 2000b):

(25) a. $\mathbf{T}x \equiv \lambda f.f\,x$
b. $\mathbf{B}fg \equiv \lambda x.f(g\,x)$
c. $\mathbf{S}fg \equiv \lambda x.fx(g\,x)$

The type-raising combinator **T** turns a term x or type σ into a function taking as its argument a function f of type $\sigma\tau$ (a function that takes a term of type σ is its argument, an abbreviated form of $\sigma{\rightarrow}\tau$) and returns a term of type τ. The result of type-raising x is therefore a function of type $(\sigma\tau)\tau$. For example, take a constant **Max** of type e. Applying **T** to this produces (among various results) functions of type $(et)t$:

(26) $\mathbf{T}(\mathbf{Max}) \equiv \lambda P_{et}.P(\mathbf{Max})$

The composition combinator **B** composes a function f of type $\sigma\tau$ with a function g of type $\delta\sigma$ before g has applied to its own argument of type δ. The result is a new function h of type $\delta\tau$ that applies the embedded function g to its argument. For example, to obtain a logical form for *Max eats*, the combinator **B** is applied to the result of (26) and to the LF for *eats*, $\lambda y\lambda z.[\mathbf{eat}(z,y)]$ (of type $e(et)$); this returns the function $\lambda x.[\mathbf{eat}(\mathbf{Max},x)]$ (of type et) as follows:

(27) $\mathbf{B}(\lambda P_{et}.[P(\mathbf{Max})])(\lambda y\lambda z.[\mathbf{eat}(z,y)]) \equiv$
$\lambda x\lambda P_{et}.[P(\mathbf{Max})](\lambda y\lambda z.[\mathbf{eat}(z,y)](x)) \equiv$
$\lambda x\lambda P_{et}.[P(\mathbf{Max})](\lambda z.[\mathbf{eat}(z,x)]) \equiv$
$\lambda x.[\mathbf{eat}(\mathbf{Max},x)]$

Finally, the substitution combinator **S** is quite similar to **B**, except that the function it creates applies both f and g to its argument x. This accounts for parasitic gap constructions such as *These are the articles which Ed filed without reading* (Steedman, 1996b). To obtain the logical form for *file without reading*, **S** is applied to the function $\lambda y\lambda z.\mathbf{file}(z,y)$ of type $e(et)$ and to another function $\lambda y\lambda Q_{et}\lambda z.[Qy \wedge \mathbf{without}(\mathbf{read}(z,y))]$ of type $e((et)(et))$. This results in a function of type eet:

(28) $\mathbf{S}(\lambda z\lambda Q_{et}\lambda y.[Qy \wedge (\mathbf{without}(\mathbf{read}(y,z)))])(\lambda z\lambda y.[\mathbf{file}(y,z)]) \equiv$
$\lambda z\lambda y.[\mathbf{file}(y,z) \wedge (\mathbf{without}(\mathbf{read}(y,z)))]$

The CCG rules corresponding to the **T**, **B** and **S** combinators are linearized versions of combinators defined over syntactic functions of the type familiar from the AB calculus, such as np, s\np or s/(s\np) according to the following principle (Steedman, 2000b):

(29) *The Principle of Combinatory Type Transparency*
All syntactic combinatory rules are type-transparent versions of one of a small number of simple semantic operations over functions.

In plainer language, syntactic rules are order-sensitive versions of the combinators defined over syntactic categories that correspond to the types over which the lambda calculus is defined. For example, if the semantic type e corresponds to the syntactic type np while the semantic type t corresponds to the syntactic type s, then the semantic type et will correspond to the two syntactic types $\mathsf{s\backslash np}$ and $\mathsf{s/np}$.

Because syntactic functions vary in terms of the linear order in which they have to combine with their arguments, each combinator corresponds to two or more syntactic rules. Just as the rule of β-reduction in the lambda calculus corresponds to both forward and backward application in the AB-calculus, there are both forward and backward versions of the type-raising, composition and substitution rules.

TYPE-RAISING: CCG employs a class of type-raising rules that mirror the effect of the combinator $\mathbf{T}$. For example, forward type-raising permits a subject noun phrase in English to become a function seeking an intransitive verb phrase:

(30) *Forward type-raising*
$$\mathsf{X} : a \quad \Rightarrow_{\mathbf{T}} \quad \mathsf{Y}/_i(\mathsf{Y}\backslash_i\mathsf{X}) : \lambda f.fa \qquad (>\mathbf{T})$$

(31) *Backward type-raising*
$$\mathsf{X} : a \quad \Rightarrow_{\mathbf{T}} \quad \mathsf{Y}\backslash_i(\mathsf{Y}/_i\mathsf{X}) : \lambda f.fa \qquad (<\mathbf{T})$$

Note that the Y in these rule definitions is a schema over types in the range of functions taking X as their domain. In other words, in the case of forward type-raising, Y can be of type s, $\mathsf{s\backslash np}$, $\mathsf{(s\backslash np)/np}$, and so on.

In the CCG literature an assumption is frequently made that the type-raising rules are not actually syntactic rules per se, meaning that they are not available for use in derivations. Instead, they are treated as operating within the lexicon and generating a family of types for each member of an atomic category.

Another assumption is that type-raised categories are used to represent the assignment of morphosyntactic information such as nominative case, agreement marking, etc. Using Modern Standard Arabic to illustrate, the NP *al-walad-u* "the boy" is marked in the nominative case and controls 3rd-person-singular agreement and might hence have the category in (32a) while *al-walad-a*, the same NP marked in the accusative case, might have the category in (32b):

(32) a. **al-walad-u** $\vdash \mathsf{(s/np_{acc})\backslash((s/np_{acc})/np_{nom,3ms})}$

b. **al-walad-a** $\vdash \mathsf{s\backslash(s/np_{acc,3ms})}$

This analysis implies a theoretical claim that noun phrases, at least those that are marked with case or which control agreement morphology, will necessarily be represented as raised categories in a syntactic derivation. This is perhaps most interesting in a language like English, where there is a *directional* difference in the categories (as opposed to just a featural difference) between nominative case (subject noun phrases, with category $\mathsf{s/(s\backslash np)}$) and non-nominitave case (e.g. object noun phrases, with category $\mathsf{s\backslash(s/np)}$).

COMPOSITION: The function composition rules add *associativity* to the AB calculus. This means that a sequence of three expressions ABC that would have

to be combined in the order A(BC) in the AB calculus can also be combined in the order (AB)C by using composition. To illustrate, consider example (6), repeated here, as it is derived in the AB calculus:

(33)
$$\dfrac{\dfrac{\textit{Ed}}{\mathsf{np}} \quad \dfrac{\dfrac{\textit{saw}}{(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}} \quad \dfrac{\textit{Ted}}{\mathsf{np}}}{\mathsf{s}\backslash\mathsf{np}}>}{\mathsf{s}}<$$

The AB-calculus provides only function application as a combinatory rule. Given the categories assigned to **Ed**, **saw** and **Ted** and the definition of function application, this means that the words have to be combined in the order **Ed**(**saw Ted**).

However, CCG provides two rules (or rather families of rules) based on the **B** combinator, forward and backward (harmonic) composition, called "harmonic" because they require that the directions of the slashes in the categories being composed are the same (and hence "harmonic"):

(34) a. *Forward harmonic composition*
$$\mathsf{X}/\mathsf{Y} \quad \mathsf{Y}/\mathsf{Z} \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}/\mathsf{Z} \qquad (>\mathbf{B})$$

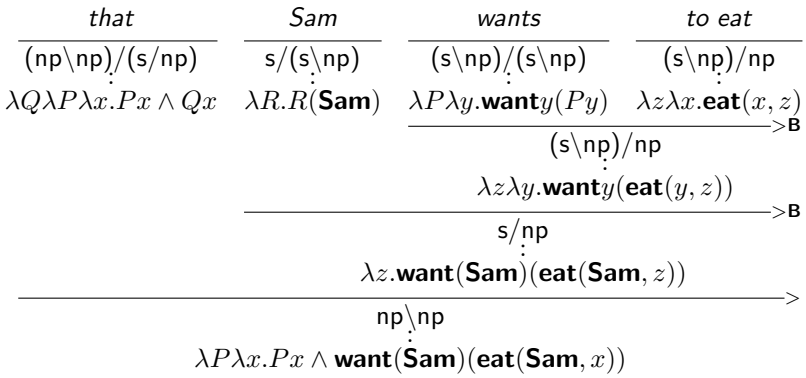
b. *Backward harmonic composition*
$$\mathsf{Y}\backslash\mathsf{Z} : g \quad \mathsf{X}\backslash\mathsf{Y} : f \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}\backslash\mathsf{Z} : \lambda x.f(gx) \qquad (<\mathbf{B})$$

Returning to the example, if **Ed** is type-raised and composed with **saw** via forward composition, the order can be reversed, allowing the words to be combined in the order (**Ed saw**)**Ted**:

(35)
$$\dfrac{\dfrac{\dfrac{\textit{Ed}}{\mathsf{np} : \mathbf{Ed}}}{\mathsf{s}/\!\!\diamond(\mathsf{s}\backslash\!\!\diamond\mathsf{np}) : \lambda P.P\mathbf{Ed}}>\mathbf{T} \quad \dfrac{\textit{saw}}{(\mathsf{s}\backslash\!\!\diamond\mathsf{np})/\!\!\diamond\mathsf{np} : \lambda x\lambda y.\mathbf{see}(y,x)} \quad \dfrac{\textit{Ted}}{\mathsf{np} : \mathbf{Ted}}}{\mathsf{s}/\!\!\diamond\mathsf{np} : \lambda x.\mathbf{see}(\mathbf{Ed},x)}>\mathbf{B}}{\mathsf{s} : \mathbf{see}(\mathbf{Ed},\mathbf{Ted})}>$$

Allowing associativity in the grammar allows elegant analyses of extraction and coordination by allowing constituents to be derived that cannot be created in the AB calculus. For example, a typical kind of extraction is relative clause formation, illustrated in the following derivation for the relative clause *that Sam wants to eat*, in which two successive applications of forward composition allow for *Sam wants to eat* to be derived as a constituent which is then taken as the argument to the relative pronoun:

(36)



A further example that cannot be handled elegantly with the limited apparatus of AB (and context-free grammar) involves the coordination of verbal complexes such as that in (37).

(37) Ed will see and should hear Ted.

A standard analysis of modal verbs is that they are functions from intransitive verb phrases into intransitive verb phrase (see Hoyt and Baldridge 2008 for a different analysis). However, for this coordination to proceed, the modal verbs *will* and *should* must combine with *see* and *hear* respectively and coordinate before their shared object argument is consumed. Because the only rule available in the AB-calculus is functional application, there is no way to do this:

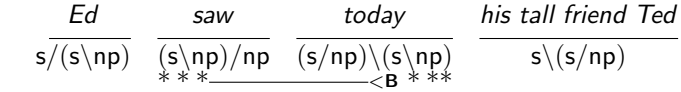
(38) $\text{will}_{(\mathsf{s}\backslash\mathsf{np})/(\mathsf{s}_{\mathsf{base}}\backslash\mathsf{np})}$ $\text{meet}_{(\mathsf{s}_{\mathsf{base}}\backslash\mathsf{np})/\mathsf{np}}$

The rule $>\mathbf{B}$, however, does allow this: for the combination in (38), X is $\mathsf{s}\backslash\mathsf{np}$, Y is $\mathsf{s}_{\mathsf{base}}\backslash\mathsf{np}$, and Z is np. The result, X/Z, is thus $(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}$.

Crossed Composition: The function application and composition rules discussed so far are all *order-preserving*, meaning that they require that syntactic functions (such as verbs) be directly adjacent to their arguments in order to be able to combine with them. Thus, they are unable to derive sequences of expressions in which function categories are not immediately adjacent to their arguments. In the CCG literature, non-adjacent arguments are said to have been *permuted*.

An example of argument permutation in English is heavy-NP shift, in which an adverb comes between a verb and its direct object, such as *Ed saw today his tall friend Ted*. Assuming that *today* is of type $(\mathsf{s}\backslash\mathsf{np})\backslash(\mathsf{s}\backslash\mathsf{np})$ and that *Ted* and *Ed* have raised categories, then the rules given so far provide no way of combining the categories in the sentence because the directions of the slashes do not match those specified in the rules:

(39)



For this reason, "non-harmonic" or "crossed" composition rules are provided:

(40) *Forward crossed composition*

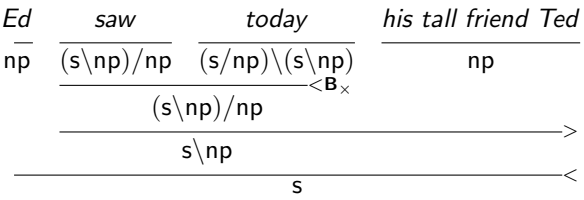
$$X/Y \quad Y\backslash Z \quad \Rightarrow_{\mathbf{B}} \quad X\backslash Z \qquad (>\mathbf{B}_{\times})$$

(41) *Backward crossed composition*

$$Y\backslash Z \quad X/Y \quad \Rightarrow_{\mathbf{B}} \quad X/Z \qquad (<\mathbf{B}_{\times})$$

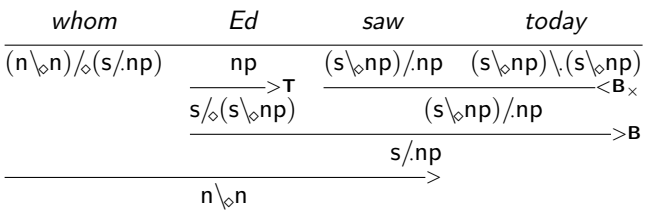
The crossed composition rules allow function categories to compose when the directions of their slashes do not match. This in turns allow argument permutation to be derived. In the example above, the backward crossed composition rule allows the adverb *today* to combine with *saw* before the verb has consumed its direct object:

(42)



The crossed composition rules also allow non-peripheral extraction to be derived:

(43) *man*

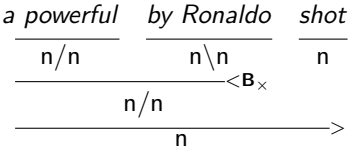


This example succinctly shows type-raising, harmonic composition, and crossed composition rules all working in concert to induce the associativity and permutativity required of the grammar.

2.2 Modalized CCG

Of course, adding the crossed composition rule is dangerous in the sense that it wildly overpredicts the acceptability of argument permutation in a language like English. For example, the forward crossed-composition rule allows the following unacceptable phrase *a powerful by Ronaldo shot* to be derived (example from Baldridge 2002):

(44)



MODALIZED SLASHES: One way of blocking derivation like this is to define language-specific rule restrictions (or outright bans) in order to limit the applicability of rules (Steedman, 2000b). This is unattractive because it means that the grammars for languages can vary both in their lexicon and their rule set. An alternative that has become standard practice in CCG is to define a set of modes of combination that can be used to create slash types which selectively license some, but not all, of a set of truly *universal* combinatory rules. Jacobson (1992b) was perhaps the first to use this strategy in a CCG-like system; there, she uses slash-types to *force* composition (e.g., for raising verbs) and disallow application. Baldridge (2002) provides a general framework for creating modalized CCG rules using underlying logics that generate the rules as proofs. Most work in CCG now uses the set of modalities $\mathcal{M} = \{\star, \diamond, \times, \cdot\}$ defined by Baldridge and Kruijff (2003). They have the following behaviors:

- $\star$: non-associative and non-commutative
- $\diamond$: associative and non-commutative
- $\times$: non-associative and commutative
- $\cdot$: associative and commutative

These modalities allow typed slashes such as $/_\star$ and $/_\diamond$ to be defined. How their behaviors are projected is explained in the remainder of this section. Their basis in Categorial Type Logics is discussed in section 3.

With slashes typed according to different modes, the rules must be defined with respect to those types. The application rules are the same as with AB, but may be used with any of the slash types, as indicated with the i subscript, where i can be any of the modalities given in $\mathcal{M}$:

(45) *Forward application*
$\mathsf{X}/_i\mathsf{Y} \quad \mathsf{Y} \quad \Rightarrow \quad \mathsf{X} \qquad (\text{for } i \in \mathcal{M}) \qquad (>)$

(46) *Backward application*
$\mathsf{Y} \quad \mathsf{X}\backslash_i\mathsf{Y} \quad \Rightarrow \quad \mathsf{X} \qquad (\text{for } i \in \mathcal{M}) \qquad (<)$

The crossed-composition rules are restricted to function categories marked with the modalities $\cdot$ and $\times$:

(47) *Forward crossed composition*
$\mathsf{X}/_i\mathsf{Y} \quad \mathsf{Y}\backslash_j\mathsf{Z} \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}\backslash_j\mathsf{Z} \qquad (\text{for } i, j \in \{\times, \cdot\}) \qquad (>\mathbf{B}_\times)$

(48) *Backward crossed composition*
$\mathsf{Y}/_j\mathsf{Z} \quad \mathsf{X}\backslash_i\mathsf{Y} \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}/_j\mathsf{Z} \qquad (\text{for } i, j \in \{\times, \cdot\}) \qquad (<\mathbf{B}_\times)$

Likewise, the harmonic composition rules are restricted to categories marked with the modalities $\cdot$ and $\diamond$:

(49) *Forward harmonic composition*
$\mathsf{X}/_i\mathsf{Y} \quad \mathsf{Y}/_j\mathsf{Z} \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}/_j\mathsf{Z} \qquad (\text{for } i, j \in \{\diamond, \cdot\}) \qquad (>\mathbf{B})$

(50) *Backward harmonic composition*

$\mathsf{Y}\backslash_j\mathsf{Z} : g \quad \mathsf{X}\backslash_i\mathsf{Y} : f \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}\backslash_j\mathsf{Z} : \lambda x.f(gx) \qquad (\text{for } i,j \in \{\diamond,\cdot\}) \quad (<\mathbf{B})$

With these modes, we can have lexical entries such as the following which allow the example of heavy-NP-shift in (42) above, but which disallow the unacceptable example in (44):

(51) a. *saw* := $(\mathsf{s}\backslash_\diamond\mathsf{np})/_\cdot\mathsf{np}$

b. *today* := $(\mathsf{s}\backslash_\diamond\mathsf{np})\backslash_\cdot(\mathsf{s}\backslash_\diamond\mathsf{np})$

c. *powerful* := $\mathsf{n}/_\star\mathsf{n}$

d. *by Ronaldo* := $\mathsf{n}\backslash_\times\mathsf{n}$

(52) a.

a	*powerful*	*by Ronaldo*	*shot*
$\mathsf{np}/_\star\mathsf{s}$	$\mathsf{n}/_\diamond\mathsf{n}$	$\mathsf{n}\backslash_\times\mathsf{n}$	n
	* * * $>_{\mathbf{B}}$	* **	

b.

Ed	*saw*	*today*	*his tall friend Ted*
np	$(\mathsf{s}\backslash\mathsf{np})/_\cdot\mathsf{np}$	$(\mathsf{s}/\mathsf{np})\backslash_\times(\mathsf{s}\backslash\mathsf{np})$	np
		$<\mathbf{B}_\times$	
	$(\mathsf{s}\backslash\mathsf{np})/_\times\mathsf{np}$		
	$\mathsf{s}\backslash\mathsf{np}$		
	s		

The advantage of using modalization of argument slashes is that it allows the rules to be universal: all the rules of modalized CCG are available to every grammar; their applicability is controlled by which modalities syntactic functions are specified with. Hence, variability in word order (or lack thereof) is a lexical property of words in a given language, rather than something that has to be stipulated in a language-specific rule set.

Extraction and coordination: The ability for subject noun phrases to combine with the verb before the verb has consumed its object noun phrases provides precisely the constituents needed for phenomena such as right-node raising, object extraction, and topicalization in English. For example, the object extraction *man whom Ed saw* is now derivable, as is unbounded object extraction. Forward composition allows the extracted argument to be successively passed up until it is revealed to the relative pronoun:

(53)

man	*whom*	*I*	*thought*	*that*	*Ed*	*saw*
n	$(\mathsf{n}\backslash_\diamond\mathsf{n})/_\diamond(\mathsf{s}/_\cdot\mathsf{np})$	np	$(\mathsf{s}\backslash_\diamond\mathsf{np})/_\diamond\mathsf{s}$	$\mathsf{s}/_\diamond\mathsf{s}$	np	$(\mathsf{s}\backslash_\diamond\mathsf{np})/_\cdot\mathsf{np}$
		$>_{\mathbf{T}}$			$>_{\mathbf{T}}$	
		$\mathsf{s}/_\diamond(\mathsf{s}\backslash_\diamond\mathsf{np})$			$\mathsf{s}/_\diamond(\mathsf{s}\backslash_\diamond\mathsf{np})$	
		$>_{\mathbf{B}}$			$>_{\mathbf{B}}$	
		$\mathsf{s}/_\diamond\mathsf{s}$			$\mathsf{s}/_\cdot\mathsf{np}$	
				$>_{\mathbf{B}}$		
				$\mathsf{s}/_\cdot\mathsf{np}$		
			$>_{\mathbf{B}}$			
			$\mathsf{s}/_\cdot\mathsf{np}$			
	$>$					
	$\mathsf{n}\backslash\mathsf{n}$					
$<$						
n						

The subject-verb constituent $s/.np$ is also implicated in right node raising:

(54)
$$\dfrac{\dfrac{\dfrac{\textit{Ed saw}}{s/.np}\quad\dfrac{\dfrac{\textit{and}}{((s/.np)\backslash_\star(s/.np))/_\star(s/.np)}\quad\dfrac{\textit{Ned heard}}{s/.np}}{(s/.np)\backslash_\star(s/.np)}>}{s/np}<\quad\dfrac{\textit{Ted}}{np}}{s}>$$

This analysis is also consistent with intonational constituency and incremental parsing with the competence grammar (Steedman, 2000b). Importantly, in addition to deriving these constituents, CCG provides compositional interpretations for them.

Other types of "odd" constituent coordinations appear cross-linguistically, such as argument cluster coordination in English: *Ed gave Ted tea and Ned bread*. This phenomenon has also been called non-constituent coordination, reflecting the difficulty in assigning a sensible phrase structure constituency that groups indirect objects with direct objects. Again, CCG's increased associativity allows a constituent to be formed in non-standard ways. To deal with such an example, we need the backward versions of the composition and type-raising rules.

These rules conspire in the analyses of Steedman (1985) and Dowty (1988) (presented in 1985) to create the necessary constituents for argument cluster coordination. After type-raising each of the two objects, they are composed, resulting in a function which is looking for a function that is missing its indirect object and direct object arguments. This function can then be coordinated with other functions of the same type. The derivations in (55-56), in which the subject has already composed with the verb, show this.

(55)
$$\dfrac{\dfrac{\dfrac{\textit{Ted}}{np}}{(s/.np)\backslash_\diamond((s/.np)/_\diamond np)}<\mathbf{T}\quad\dfrac{\dfrac{\textit{tea}}{np}}{s\backslash.(s/.np)}<\mathbf{T}}{s\backslash_\diamond((s/.np)/_\diamond np)}<\mathbf{B}\quad\dfrac{\dfrac{\dfrac{\textit{and}}{(X\backslash_\star X)/_\star X}\quad\dfrac{\dfrac{\textit{Ned}}{np}}{(s/.np)\backslash_\diamond((s/.np)/_\diamond np)}<\mathbf{T}\quad\dfrac{\dfrac{\textit{bread}}{np}}{s\backslash.(s/.np)}<\mathbf{T}}{s\backslash_\diamond((s/.np)/_\diamond np)}<\mathbf{B}}{(s\backslash_\diamond((s/.np)/_\diamond np))\backslash_\star(s\backslash_\diamond((s/.np)/_\diamond np))}>}{s\backslash_\diamond((s/.np)/_\diamond np)}<$$

(56)
$$\dfrac{\dfrac{\textit{Ed gave}}{(s/.np)/_\diamond np}\quad\dfrac{\textit{Ted tea and Ned bread}}{s\backslash_\diamond((s/.np)/_\diamond np)}}{s}<$$

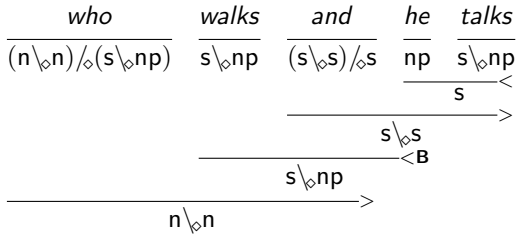
Because the combinatory rules are semantically consistent, this derivation produces a meaning representation of the following form:

(57) $\mathbf{give}(\mathbf{Ed}, \mathbf{Ted}, \mathbf{tea}) \wedge \mathbf{give}(\mathbf{Ed}, \mathbf{Ned}, \mathbf{bread})$,

This captures the correct dependencies between the verbal predicate and its arguments.

With the availability of the composition rules in the universal grammar, it is important that the coordinating categories have the most restrictive $\star$ slashes used thus far. If they were given instead categories of the form $(X\backslash_{\diamond}X)/_{\diamond}X$, it would be possible to produce the following undesirable derivation:

(58) **man*



With the restrictive category $(s\backslash_{\star}s)/_{\star}s$, the composition of *walks* with *and he talks* is blocked because $<\mathbf{B}$ is only defined for the modalities $\diamond$ and $\cdot$, so *and he talks* with category $s\backslash_{\star}s$ cannot compose with $s\backslash_{\diamond}np$ of *walks*.

SUBSTITUTION: CCG makes available only one further rule class, based on the substitution combinator **S** (Szabolcsi, 1987). The combinator **S** is different from **B** and **T** in that it allows a single resource to be utilized by two different functors. We find the need for such a combinator in parasitic gap constructions (Ross, 1967) such as the following:

(59) Ed copied and filed without reading, your handbook article.

(60) articles which Ed filed without reading

In both examples, a single dependent acts as the argument of both *filed* and *without reading*. If we consider the categories of the constituents, it is clear that the rules defined thus far will not allow the derivation to proceed.

(61) $\text{filed}_{(s\backslash_{\diamond}np)/_{\cdot}np}$ $[\text{without reading}]_{((s\backslash_{\diamond}np)\backslash_{\cdot}(s\backslash_{\diamond}np))/_{\cdot}np}$

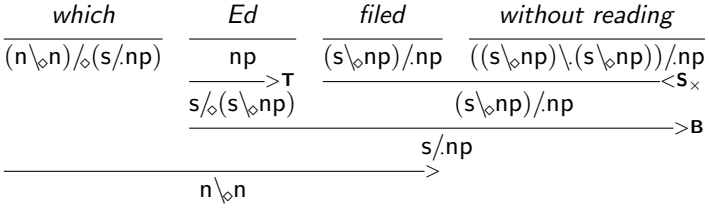
The fact that sequences like *filed without reading* can coordinate with transitive verbs as in the example indicates that it must be possible to combine their categories to form a transitive category. The following rule provides exactly this functionality:

(62) *Backward crossed substitution*

$$Y/_jZ \quad (X\backslash_iY)/_jZ \quad \Rightarrow_{\mathbf{S}} \quad X/_jZ \qquad (\text{for } i,j \in \{\times,\cdot\}) \qquad (<\mathbf{S}_{\times})$$

With this rule available in the system, the derivation of (60) is:

(63) *articles*



Naturally, there are two other harmonic rules and a forward crossed rule based on the substitution combinator. For further discussion on the substitution rules and a more detailed account of parasitic gaps in English, see Steedman (1996a).

Other combinators: As CCG analyses are performed for a wider range of linguistic data, certain data types come up which the rules discussed above are not able to derive. Several kinds of such data involve pronominal binding and are discussed in the following subsection. Another kind involves sentences like the following, discussed by Hoyt and Baldridge (2008).

(64) Make sure you also read the instructions about **what you can** and **what you should NOT** upload to this website.

The problem that this example presents (see Hoyt and Baldridge for further examples from several languages) is that the sequences *what you can* and *what you should not* have to be put together as constituents before they can be combined with *and*:

(65)
$$\frac{\dfrac{\textit{what}}{\mathsf{s/(s/np)}} \quad \dfrac{\dfrac{\textit{you can}}{\mathsf{s/(s\backslash np)}} \quad \dfrac{\textit{and}}{\mathsf{(X\backslash X)/X}} \quad \dfrac{\textit{what}}{\mathsf{s/(s/np)}} \quad \dfrac{\textit{you should not}}{\mathsf{s/(s\backslash np)}}}{\mathsf{s/(s/np)}}\,{>}\mathbf{B}}{\mathsf{s/(s/np)}}\,{>}\mathbf{B}$$

Hoyt and Baldridge propose the following rules for deriving examples like this, based on proofs in an underlying categorial type logic:

(66) $\mathsf{X/_\diamond(Y/_\diamond Z)} : f \quad \mathsf{Y/_\diamond W} : g \quad \Rightarrow_{\mathbf{S}} \quad \mathsf{X/_\diamond(W/_\diamond Z)} : \lambda h.f(\lambda x.ghx) \qquad (>\mathbf{D})$

(67) $\mathsf{Y\backslash_\diamond W} : f \quad \mathsf{X\backslash_\diamond(Y\backslash_\diamond Z)} : g \quad \Rightarrow_{\mathbf{S}} \quad \mathsf{X\backslash_\diamond(W\backslash_\diamond Z)} : \lambda h.f(\lambda x.ghx) \qquad (<\mathbf{D})$

(68)
$$\frac{\dfrac{\dfrac{\textit{what}}{\mathsf{s/(s/np)}} \quad \dfrac{\textit{you can}}{\mathsf{s/(s\backslash np)}}}{\mathsf{s/((s\backslash np)/np)}}\,{>}\mathbf{D} \qquad \dfrac{\dfrac{\textit{and}}{\mathsf{(X\backslash X)/X}} \quad \dfrac{\dfrac{\textit{what}}{\mathsf{s/(s/np)}} \quad \dfrac{\textit{you should not}}{\mathsf{s/(s\backslash np)}}}{\mathsf{s/((s\backslash np)/np)}}\,{>}\mathbf{D}}{\mathsf{((s\backslash np)/np)\backslash((s\backslash np)/np)}}\,{>}}{\mathsf{(s\backslash np)/np}}\,{<}$$

Future research will perhaps reveal additional kinds of examples that the current CCG rule set will not account for.

2.3 Further combinators

The Division Combinator: Another variation on CCG is developed by Pauline Jacobson in a sequence of carefully argued papers (Jacobson, 1990, 1992a, 1993, 1999, 2000, 2003, a.o.) in which she analyzes data involving intricate binding interpretations. The version of categorial grammar that she develops differs from the version of CCG discussed above (referred to here for convenience as "BTS-CCG") in making central use of what she (and others) refer to as the "Geach Rule." It is otherwise known as the *Division* combinator (referred to

here as the **G**-rule), which is equivalent to a unary version of the **B**-combinator discussed above.[2]

(69) *Forward Division*
$\mathsf{X/Y} : f_{\sigma\tau} \quad \Rightarrow_{>\mathbf{G}} \quad (\mathsf{X}\!\nwarrow\!\mathsf{Z})/(\mathsf{Y}\!\nwarrow\!\mathsf{Z}) : \lambda P_{\delta\sigma}\lambda x_{\delta}.f(P(x)) \qquad (> \mathbf{G})$

(70) *Backward Division*
$\mathsf{X\backslash Y} : f_{\sigma\tau} \quad \Rightarrow_{<\mathbf{G}} \quad (\mathsf{X}\!\nwarrow\!\mathsf{Z})\backslash(\mathsf{Y}\!\nwarrow\!\mathsf{Z}) : \lambda P_{\delta\sigma}\lambda x_{\delta}.f(P(x)) \qquad (< \mathbf{G})$

In addition to the **G**-rules, Jacobson also assumes the familiar type-raising (**T**) rules as well as two further rules to provide analyses for a variety of subtle binding data. Rule (71a) is a unary version of the substition rules and is used to model pronominal binding. Rule (71b) is a "de-Curry-ing" rule which changes a function of type $\sigma(\delta\tau)$ to a function of type $(\sigma\delta)\tau$. Jacobson uses the M-rule to model functional interpretations of relative clauses (Jacobson, 2000, 2002).

(71) a. *Rule Z*
$(\mathsf{X\backslash Y})/\mathsf{Z} : \lambda y\lambda x.fxy \quad \Rightarrow_{Z} \quad (\mathsf{X\backslash Y})/(\mathsf{Z}\!\nwarrow\!\mathsf{Z}) : \lambda g\lambda x.f(gx)x) \qquad (Z)$

b. *Rule M*
$(\mathsf{X|Y})|\mathsf{Z} : \lambda y\lambda x.fxy \quad \Rightarrow_{M} \quad \mathsf{X}|(\mathsf{Y}|\mathsf{Z}) : \lambda g.\forall x[x{\in}dom(g) \rightarrow fxgx] \quad (M)$

Jacobson treats pronouns as semantic identity functions $\lambda x.x$ (of type ee) and syntactically as functions of category $\mathsf{np}\!\nwarrow\!\mathsf{np}$. Binding is realized by means of the Z-rule, which turns predicate categories into categories taking as arguments functions what we will call "pronominal functions," namely functions with the $\nwarrow$-operator looking for an np argument.

To illustrate with a simple example, consider the sentence *Every man$_i$ thinks Mary loves him$_i$*, interpreted with the quantifier *every man* binding the pronoun *him*. This is derived in (72), with successive application of the G-rule to the categories above the pronoun in the binding dependency. The type-changing rule Z changes the type for *think* into one taking a pronominal function as its argument.

(72)

thinks	*Mary*	*loves*	*him*
$(\mathsf{s\backslash np})/\mathsf{s}$	$\mathsf{s}/(\mathsf{s\backslash np})$	$(\mathsf{s\backslash np})/\mathsf{np}$	$\mathsf{np}\!\nwarrow\!\mathsf{np}$
$\lambda p.\lambda y.think'yp$	$\lambda Q_{et}.Q(Mary')$	$\lambda z.\lambda x.love'xz$	$\lambda z.z$
$(\mathsf{s\backslash np})/(\mathsf{s}\!\nwarrow\!\mathsf{np})\ Z$	$(\mathsf{s}\!\nwarrow\!\mathsf{np})/((\mathsf{s\backslash np})\!\nwarrow\!\mathsf{np})\ G$	$((\mathsf{s\backslash np})\!\nwarrow\!\mathsf{np})/(\mathsf{np}\!\nwarrow\!\mathsf{np})\ G$	
$\lambda R_{et}.\lambda x.think'x(Rx)$	$\lambda R_{eet}.\lambda z.R(Mary')z$	$\lambda P_{ee}.\lambda z.\lambda x.love'x(Pz)$	
		$(\mathsf{s\backslash np})\!\nwarrow\!\mathsf{np} : \lambda z.\lambda x.love'xz$ >	
	$\mathsf{s}\!\nwarrow\!\mathsf{np} : \lambda z.love'(Mary')z$ >		
$\mathsf{s\backslash np} : \lambda x.think'x(love'(Mary')x)$ >			

[2]Note that in Jacobson's notation, the category $\mathsf{X^Y}$ is a function type, while here we represent this type using an $\nwarrow$-operator, as in $\mathsf{X}\!\nwarrow\!\mathsf{Y}$.

The quantifier *every man* then takes this expression as its argument, binding the pronoun indirectly.[3]

(73)
$$\dfrac{\dfrac{\textit{every man}}{\mathsf{s}/(\mathsf{s}\backslash\mathsf{np}) : \lambda P_{et}.\forall x[man'x \rightarrow Px]} \quad \dfrac{\textit{thinks Mary loves him}}{\mathsf{s}\backslash\mathsf{np} : \lambda x.think'x(love'(Mary')x)}}{\mathsf{s} : \forall x[man'x \rightarrow think'x(love'(Mary')x)]}\,{>}$$

Note that Jacobson's analysis of pronominal binding can be recreated within BTS-CCG, given that the combination of the **G**-rule and function application corresponds directly to function composition (**B**) in BTS-CCG. Likewise, the Z-rule can be assumed to be a lexical rule, in line with the standard treatment of unary type-changing rules in the BTS-CCG literature. The binding effect in (72) can then be captured in BTS-CCG as follows (we subscript slash-operators corresponding to the $\backslash$operator above with a "g"):

(74)
$$\dfrac{\dfrac{\dfrac{\textit{thinks}}{(\mathsf{s}\backslash\mathsf{np})/(\mathsf{s}/_{\mathsf{g}}\mathsf{np})} \quad \dfrac{\textit{Mary}}{\mathsf{s}/(\mathsf{s}\backslash\mathsf{np})} \quad \dfrac{\dfrac{\textit{loves}}{(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}} \quad \dfrac{\textit{him}}{\mathsf{np}/_{\mathsf{g}}\mathsf{np}}}{(\mathsf{s}\backslash\mathsf{np})/_{\mathsf{g}}\mathsf{np} : \lambda z.\lambda x.love'xz}\,{>}\mathbf{B}}{\mathsf{s}/_{\mathsf{g}}\mathsf{np} : \lambda z.love'(Mary')z}\,{>}\mathbf{B}}{\mathsf{s}\backslash\mathsf{np} : \lambda y.think'y(love'(Mary')y)}\,{>}$$

From this perspective, analyses in G-CCG can be straightforwardly captured in BTS-CCG, and vice versa.

However, one potential point of difference between the two formulations of CCG involves locality restrictions on application of the rules. As was discussed above, BTS-CCG assumes that application of the harmonic and composition rules are restrained by the modalities. Baldridge (2002) argues that restrictions on extraction out of adjuncts and coordinate structures be modeled in terms of modal restrictions. In particular, he argues that adjuncts such as relative clauses as well as *and* have types specified with the $\star$-modality:

(75) *that, which* $\vdash (\mathsf{np}\backslash\mathsf{np})/_{\star}(\mathsf{s}/\mathsf{np}) : \lambda Q_{et}.\lambda P_{et}.\lambda x.[Px \wedge Qx]$

(76) *and* $\vdash (\mathsf{X}\backslash_{\star}\mathsf{X})/_{\star}\mathsf{X} : \lambda q_{\sigma\tau}.\lambda p_{\sigma\tau}.\lambda x_{\sigma}.[p(x) \wedge q(x)]$

The presence of the $\star$-modality on these categories prevents the composition rules from applying to them and so prevents extraction dependencies from being established from within their arguments.

Jacobson, however, used her framework to analyze data showing binding from within conjunct (77) and adjunct islands (78):

(77) Every man$_i$ thinks that [he$_i$ lost and Mary won].

(78) Every man$_i$ hopes that the woman [that he$_i$ wants to marry] loves him.

[3]See Szabolcsi (2003) for an extension of Jacobson's framework to cross-sentential anaphora.

For example, in Jacobson's framework the coordinate structure in (77) would be analyzed as follows.

(79)

he	*lost*	*and*	*Mary*	*won*
$\mathsf{np}\uparrow\mathsf{np}$	$\mathsf{s}\backslash\mathsf{np}$	$(\mathsf{s}\backslash\mathsf{s})/\mathsf{s}$	np	$\mathsf{s}\backslash\mathsf{np}$
$\lambda x.x$	$\lambda y.lose'y$	$\lambda q.\lambda p.p \wedge q$	$Mary'$	$\lambda z.win'z$
	$(\mathsf{s}\uparrow\mathsf{np})\backslash(\mathsf{np}\uparrow\mathsf{np})$ (G)			s (<)
	$\lambda P_{ee}.\lambda y.lose'(Py)$			$win'(Mary')$
	$\mathsf{s}\uparrow\mathsf{np} : \lambda y.lose'y$ (<)			$\mathsf{s}\backslash\mathsf{s} : \lambda p.p \wedge win'(Mary')$ (>)
				$(\mathsf{s}\uparrow\mathsf{np})\backslash(\mathsf{s}\uparrow\mathsf{np}) : \lambda Q_{et}.\lambda x.Qx \wedge win'(Mary')$ (G)
$\mathsf{s}\uparrow\mathsf{np} : \lambda x.lose'x \wedge win'(Mary')$ (<)				

Reproducing this analysis in BTS-CCG requires function composition into the right conjunct of the coordinate structure. Assuming that *and* has the type $(\mathsf{s}\backslash_\star\mathsf{s})/_\star\mathsf{s}$, this is impossible.

(80)

he	*lost*	*and*	*Mary*	*won*
$\mathsf{np}\uparrow\mathsf{np}$	$\mathsf{s}\backslash\mathsf{np}$	$(\mathsf{s}\backslash_\star\mathsf{s})/_\star\mathsf{s}$	np	$\mathsf{s}\backslash\mathsf{np}$
$\lambda x.x$	$\lambda y.lose'y$	$\lambda q.\lambda p.p \wedge q$	$Mary'$	$\lambda z.win'z$
$\mathsf{s}\uparrow\mathsf{np} : \lambda y.lose'$ (<B)			$\mathsf{s} : win'(Mary')$ (<)	
		$\mathsf{s}\backslash_\star\mathsf{s} : \lambda p.p \wedge win'(Mary')$ (>)		
* * * (<B$_\times$)				* **

Therefore, interpreting Jacobson's analysis in BTS-CCG by reducing the G-rule + FA to function composition loses coverage for examples like this. By the same token, the Geach framework that Jacobson develops in her papers provides no means for imposing restrictions on the application of the G-rule. If it were to be used to model extraction as well as pronominal binding, it would therefore fail to account for adjunct- and coordinate structure constraints that BTS-CCG captures.

Note, however, that this difference in coverage is a technical matter and reflects the different research interests in the two communities. Much of the work in BTS-CCG focuses on modeling extraction dependencies and other kinds of word order permutatation. As a consequence, the modeling of extraction restrictions has been a more important concern in the BTS-CCG literature. Jacobson, on the other hand, has focused almost entirely on pronominal binding phenomena and has had little to say about modeling extraction dependencies (Jacobson, 1990). Reconciling the two approaches therefore is simply a matter of technical innovation. As discussed in the next section, the modalized form of CCG is actually based on a particular set of structural rule postulates in CTL. This core logic can be seen as a generator of rules for CCG, and indeed, the Geach rule, among others, are theorems of that logic (Baldridge, 2002). The modalities for these rules are ensured to be consistent, so they would enforce the same constraints for a Geach-based derivation like (79) as they would for

one based on the standard CCG combinators like (80). This points to a way to reconcile the empirical coverage of the two approaches: augment the underlying logic so that the $\{\times, \diamond, \star, \cdot\}$ modality set used in BTS-CCG has an additional modality (say, $+$) that allows composition with any of the other modalities. This would be done via structure rule postulates (see next section) that generate rules such as the following.

(81) *Anaphoric Composition*

$$\mathsf{X}/_i\mathsf{Y} \quad \mathsf{Y}\backslash_+\mathsf{Z}_{+\mathsf{pro}} \quad \Rightarrow_{>\mathsf{B}_+} \quad \mathsf{X}\backslash_+\mathsf{Z}_{+\mathsf{pro}} \qquad (\text{for } i \in \{\star, \times, \diamond, \cdot\}) \qquad (>\mathsf{B}_+)$$

For the sake of notational felicity, we abbreviate $\backslash_+$ with the $\nwarrow$-symbol used above: $\backslash_+ = \nwarrow$.

With this additional rule, Jacobson's examples can be derived while still retaining the precise modal control over long-distance dependencies that BTS-CCG offers.

(82)

thinks	*he*	*lost*	*and*	*Mary*	*won*
$(\mathsf{s}\backslash\mathsf{np})/(\mathsf{s}\nwarrow\mathsf{np})$	$\mathsf{np}\nwarrow\mathsf{np}$	$\mathsf{s}\backslash\mathsf{np}$	$(\mathsf{s}\backslash_\star\mathsf{s})/_\star\mathsf{s}$	np	$\mathsf{s}\backslash\mathsf{np}$
$\lambda P_{et}.\lambda x.think'x(Px)$	$\lambda x.x$	$\lambda y.lose'y$	$\lambda q.\lambda p.p \wedge q$	$Mary'$	$\lambda z.win'z$
	$<\mathsf{B}_+$: $\mathsf{s}\nwarrow\mathsf{np} : \lambda\mathsf{y}.\mathsf{lose}'\mathsf{y}$			$<$: $\mathsf{s} : win'(Mary')$	
			$>$: $\mathsf{s}\backslash_\star\mathsf{s} : \lambda p.p \wedge win'(Mary')$		
		$<\mathsf{B}_+$: $\mathsf{s}\nwarrow\mathsf{np} : \lambda y.lose'y \wedge win'(Mary')$			
$>$: $\mathsf{s}\backslash\mathsf{np} : \lambda x.think'x(lose'x \wedge win'(Mary'))$					

Augmenting BTS-CCG with this rule would make it possible to capture other kinds of binding relationships that have not received much attention in the framework, such as the extensive use of resumptive pronouns in languages like Arabic, Hebrew, and others (Demirdache, 1991, 1997; Aoun and Benmamoun, 1998; Aoun and Choueiri, 2000; Aoun *et al.*, 2001; Ouhalla, 2001; Choueiri, 2002; Aoun and Li, 2003; Asudeh, 2005, 2012; Hoyt, 2010, a.o.).[4]

Other technical innovations could be relevant for such problems, especially "hat categories" that allow a form of lexically encoded type-changing (Honnibal, 2009; Honnibal and Curran, 2009).

The Wrap Rule: Other combinators have been considered in other contexts, most notably the commuting combinator ($\mathbf{C}fxy \equiv fyx$) in the guise of the wrap rule (Bach, 1979). Most notably, wrap rules have been employed by much work by Dowty (e.g., 1982; 1997). One of the uses of wrap rules is that they allow verbal categories to be defined according to a hierarchy of grammatical functions, where each of the arguments corresponding to those functions is represented by adding it to the subcategorization based on the previous function (Dowty, 1997). For example, a *subject* is a noun phrase that combines with s/np (or $\mathsf{s}\backslash\mathsf{np}$) to form an s, an *object* is a noun phrase that combines with $(\mathsf{s}/\mathsf{np})/\mathsf{np}$

[4]For an analysis of anaphora in a CTL-framework, see Hepple (1990); Jäger (1996, 1997, 2001, 2005).

(or $(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}$) to form s/np (or $\mathsf{s}\backslash\mathsf{np}$), and so on. Wrap-enabling slashes then take care of mismatches in word order.

Concretely, consider a verb-initial language, where the intransitive would be $\mathsf{s}/\mathsf{np}_\mathsf{s}$. According to Dowty's recipe for constructing verbal categories, this would lead to the transitive category $(\mathsf{s}/\mathsf{np}_\mathsf{s})/\mathsf{np}_\mathsf{o}$, which produces VOS word order. For VSO languages, Dowty takes the category to be $(\mathsf{s}/\mathsf{np}_\mathsf{s})/\!\!/_w\mathsf{np}_\mathsf{o}$, where the $/\!\!/_w$ slash does not produce adjacent concatenation of the verb and the object, but instead "wraps" them around the as-yet-unconsumed subject. In CCG, this effect could be achieved with a unary rule that reorders such categories:

(83) *Forward Wrap*
$(\mathsf{X}/\mathsf{Y})/\!\!/_w\mathsf{Z} \quad \Rightarrow_{Wrap} \quad (\mathsf{X}/\mathsf{Z})/\mathsf{Y} \qquad (> W)$

However, in CCG, the effect of such a rule is assumed to be a process that could be carried out in the lexicon (Steedman and Baldridge, 2011), meaning that one could define category hierachies in the lexicon in the manner suggested by Dowty, but still use the linearized category $(\mathsf{s}/\mathsf{np}_\mathsf{o})/\mathsf{np}_\mathsf{s}$ for VSO verbs in syntax. Dowty provides a different characterization of wrap in terms of commutative rules in multi-modal type-logical grammar (see the next section).

One of the punchlines of the wrap story is that the binding theory can be defined in terms of the structures defined by categories themselves and the derivations they license. This contrasts with Steedman's account of binding in CCG, which is based on *c*-command defined over logical forms (Steedman, 2000a). Dowty suggests that this use of logical form is at odds with Montogovian assumptions and should thus be dispreferred to the wrap analysis. Steedman argues that binding of true reflexives and bound anaphors is a strictly clause bounded phenomena, and thus constraints on binding may be specified in the lexicon. Furthermore, he suggests that whether this is done in terms of constraints over derivation structures (categories) or a logical form corresponding to them is immaterial: whereas Dowty makes no intrinsic use of logical form, Steedman makes no intrinsic use of derivation structure.

3 Categorial Type Logic

CCG adds to the AB calculus a finite set of rules of category combination. These rules selectively introduce more inference patterns that support greater associativity and/or commutativity. An alternative is to define a fully logical characterization of categorial grammar. To begin on this path, note first the similarity of the rules of the AB calculus with *Modus Ponens*: both types of rules involve the use of inference. However, in the forward application rule ($\mathsf{X}/\mathsf{Y}\ \mathsf{Y} \Rightarrow \mathsf{X}$), the resource Y is *eliminated*. This suggests that the AB calculus is related to resource-sensitive linear logic (Girard, 1987). Viewed this way, the AB calculus is a system of inference; however, unlike linear logic, the AB calculus is a partial system of inference because it has no corresponding ability to *introduce* resources: in other words, it does not support hypothetical reasoning. Furthermore, linear logic also allows different types of logical connectives to

be defined; each connective may exhibit different behaviors with respect to associativity and commutativity (and other dimensions).

The characterization of categorial grammar as a logic was first proposed by Lambek (1958) and has been since developed into the framework known as Categorial Type Logics (CTL, a.k.a. Type-Logical Grammar and subsuming various forms of the Lambek calculus) (Morrill, 1994; Hepple, 1995; Carpenter, 1998; Moortgat, 1999; Oehrle, 2011). CTL supports multiple unary and binary modes of grammatical combination that can exhibit quite different logical properties, which in turn lead to different syntactic possibilities (Moortgat and Oehrle, 1994).

Residuation: From the perspective of CTL, slashes are directionally sensitive implications, in the full logical sense of the term. As with CCG, slashes may be typed, where each type corresponds to a particular mode of combination with different logical properties (e.g., associativity, commutativity). Each slash pair $\backslash_i$, $/_i$ is related to a product operator $\bullet_i$; the three operators are related by the residuation laws:

(84) $\mathsf{A} \vdash \mathsf{C}/_i\mathsf{B}$ *iff* $\mathsf{A}\bullet_i\mathsf{B} \vdash \mathsf{C}$ *iff* $\mathsf{B} \vdash \mathsf{C}\backslash_i\mathsf{A}$

These laws say that if one can conclude the formula C when the formula A is next to the formula B (i.e., $\mathsf{A}\bullet_i\mathsf{B}$), then one also knows that from A it is possible to conclude $\mathsf{C}/_i\mathsf{B}$ and that likewise from B to conclude $\mathsf{C}\backslash_i\mathsf{A}$. The slash $\backslash_i$ is the right residual of $\bullet_i$, and $/_i$ is the left residual. This may not seem particularly intuitive, but as an analogy, think of multiplication: from $A \times B = C$ one knows that $A = \frac{C}{B}$ and $B = \frac{C}{A}$. Because multiplication is commutative, the operator $\div$ is both the left and right residual of $\times$.

In linguistic terms, a category formed with the product like $\mathsf{np}\bullet_\star\mathsf{np}$ represents the juxtaposition of two noun phrases under the modality $\star$. Slash categories are the same as with CCG: incomplete expressions seeking other expressions. A category that uses all of the operators is $(\mathsf{s}\backslash_\diamond\mathsf{np})/_\diamond(\mathsf{np}\bullet_\star\mathsf{np})$, which would be a candidate category for ditransitive verbs in English. Nonetheless, because the actual linguistic use of product is fairly rare in the CTL literature, the rules for using it are omitted in this discussion. See Moortgat (1997) or Vermaat (2005) for further details on the use of product.

Base Logic: The core of a CTL system is a universal component, referred to as the *base logic*, which defines the basic behavior common to all of the connectives. From the perspective of the AB calculus, the most familiar parts of the base logic are the slash elimination rules. These correspond to AB's application rules, but there are important differences. Most importantly, slash elimination is not necessarily tied to string adjacency as it is in functional application. Instead, these rules include a structure building component that organizes the antecedents of the premises into a new structured antecedent (which may be subsequently restructured and reordered).[5]

[5]The reader should note that in the CTL literature two notational conventions are used for representing categories and proofs. One is referred to as the natural deduction presentation and more closely resembles the notation used for CCG categories and derivations and for which

(85) *Slash elimination schemas* (with $i \in \mathcal{M}$):

a. $$\frac{\Gamma \vdash \mathsf{X}/_i\mathsf{Y} \qquad \Delta \vdash \mathsf{Y}}{(\Gamma \ \circ_i \ \Delta) \vdash \mathsf{X}}\,[/_iE]$$

b. $$\frac{\Delta \vdash \mathsf{Y} \qquad \Gamma \vdash \mathsf{X}\backslash_i\mathsf{Y}}{(\Delta \ \circ_i \ \Gamma) \vdash \mathsf{X}}\,[\backslash_iE]$$

Note how the direction of the slash is reflected in the order of the components of the structured antecedent.

With these rules, we can provide the following proof that the sentence *Ed saw Ted today* is of type s:

(86)
$$\dfrac{\mathit{Ed} \vdash \mathsf{np} \qquad \dfrac{\dfrac{\mathit{saw} \vdash (\mathsf{s}\backslash_\diamond\mathsf{np})/\!\cdot\mathsf{np} \quad \mathit{Ted} \vdash \mathsf{np}}{(\mathit{saw} \ \circ_\cdot \ \mathit{Ted}) \vdash \mathsf{s}\backslash_\diamond\mathsf{np}}\,[/E] \qquad \mathit{today} \vdash (\mathsf{s}\backslash_\diamond\mathsf{np})\backslash\!\cdot(\mathsf{s}\backslash_\diamond\mathsf{np})}{((\mathit{saw} \ \circ_\cdot \ \mathit{Ted}) \ \circ_\cdot \ \mathit{today}) \vdash \mathsf{s}\backslash_\diamond\mathsf{np}}\,[\backslash E]}{(\mathit{Ed} \ \circ_\diamond \ ((\mathit{saw} \ \circ_\cdot \ \mathit{Ted}) \ \circ_\cdot \ \mathit{today})) \vdash \mathsf{s}}\,[\backslash_\diamond E]$$

This is the standard presentation format for CTL analyses. Note that the structured antecedents contain lexical items, but these are actually standing in for the categories that they licensed from the lexicon. This greatly enhances readability, but readers should keep in mind that the categories are in the antecedents. This matters for a number of ways in which structures are manipulated, starting with hypothetical reasoning, which we turn to next.

Proof (86) only builds structure, using the elimination rules. However, the base logic also supports hypothetical reasoning: hypothesized elements can be consumed during the course of a proof and thus become part of the structured antecedent built during the process. In order for the proof to succeed, these hypotheses must eventually be discharged. This requires them to be on the periphery of the structured antecedent in order for the slash introduction rules (87) to apply. A hypothesized element on the right periphery may be discharged with a rightward slash (87a), and one on the left periphery may be elimanated with a leftward slash (87b).

(87) *Slash introduction schemas*:

a. $$\frac{\begin{array}{c}\cdots\,[y \vdash \mathsf{Y}]\\ \vdots \\ (\Gamma \ \circ_i \ y) \ \vdash \mathsf{X}\end{array}}{\Gamma \vdash \mathsf{X}/_i\mathsf{Y}}\,[/_iI]$$

reason it is used here. The other notation is referred to as the (Gentzen) sequent presentation. See (Carpenter, 1998, Ch.5) for an accessible introduction to the two presentations.

b.
$$\dfrac{\begin{array}{c}[y \vdash \mathsf{Y}]\cdots \\ \vdots \\ (y \ \circ_i \ \Gamma) \vdash \mathsf{X}\end{array}}{\Gamma \vdash \mathsf{X}\backslash_i\mathsf{Y}}\,[\backslash_i I]$$

Note that just as eliminating a slash with a modality i builds structure by connecting two antecedents with $\circ_i$, an introduced slash inherits its modality from the structure which produced it.

Interestingly, a consequence of hypothetical reasoning combined with slash introduction is that the type-raising rules given in the previous section are *theorems* of the base logic. This is shown by hypothesizing a function which consumes the argument, and then subsequently withdrawing the assumption.

(88)
$$\dfrac{\dfrac{\mathit{Ed} \vdash \mathsf{np} \qquad [x_1 \vdash \mathsf{s}\backslash_\diamond\mathsf{np}]^1}{(\mathit{Ed} \ \circ_\diamond \ x_1) \vdash \mathsf{s}}\,[\backslash_\diamond E]}{\mathit{Ed} \vdash \mathsf{s}/_\diamond(\mathsf{s}\backslash_\diamond\mathsf{np})}\,[/_\diamond I]^1$$

The reasoning in this proof, in words, is that *if* one had an intransitive verb, *then* it could consume the np (introduced by Ed from the lexicon), and derive an s. But the assumed verb x_1 *must* then be withdrawn. This introduces the rightward slash (since x_1 is on the right in the structured antecedent) with the intransitive verb category as the argument. Note that the hypotheses are marked by numbers when they are introduced and discharged to improve readability of proofs which have multiple hypothesized elements.

STRUCTURAL REASONING: The base logic still has a fairly hands-off approach to the structured antecedents of proof terms and as such it is not more flexible than the AB calculus. However, it is possible to augment the base logic by defining *structural rules* that reconfigure the antecedent set of premises and thereby create systems with varying levels of flexibility. For example, the following rules will permit structures built via the modalities $\diamond$ and $\cdot$ to be associatively restructured.

(89) *Right Association*:
$$\dfrac{(\Delta_a \ \circ_i \ (\Delta_b \ \circ_j \ \Delta_c)) \vdash \mathsf{X}}{((\Delta_a \ \circ_i \ \Delta_b) \ \circ_j \ \Delta_c) \vdash \mathsf{X}}\,[RA] \qquad (\text{for } i,j \in \{\diamond,\cdot\})$$

(90) *Left Association*:
$$\dfrac{((\Delta_a \ \circ_i \ \Delta_b) \ \circ_j \ \Delta_c) \vdash \mathsf{X}}{(\Delta_a \ \circ_i \ (\Delta_b \ \circ_j \ \Delta_c)) \vdash \mathsf{X}}\,[LA] \qquad (\text{for } i,j \in \{\diamond,\cdot\})$$

Rules such as this are one component for providing the flexibility which the AB calculus lacks and which CCG gains with rules based on combinators. For example, consider the object relative clause *man whom Ed saw*, using the same categories assumed in the CCG derivation (53). We begin by introducing entries from the lexicon and hypothesizing the missing object of *saw*. We then combine the premises using the slash elimination rules of the base logic, and restructure the binary tree built up during the proof using the structural rule of right association. Finally, we discharge the assumption using the rightward slash introduction rule, leaving the category s/np required by the relative pronoun.

(91)
$$\dfrac{\mathit{man} \vdash \mathsf{n} \qquad \dfrac{\mathit{whom} \vdash (\mathsf{n}\backslash_\diamond\mathsf{n})/_\diamond(\mathsf{s}/\mathsf{np}) \qquad \dfrac{(\mathit{Ed} \circ_1 \mathit{saw}) \vdash \mathsf{s}/\mathsf{np}}{((\mathit{Ed} \circ_\diamond \mathit{saw}) \circ x_1) \vdash \mathsf{s}}[/I]^1}{(\mathit{whom} \circ_\diamond (\mathit{Ed} \circ_\diamond \mathit{saw})) \vdash \mathsf{n}\backslash_\diamond\mathsf{n}}[/_\diamond E]}{(\mathit{man} \circ_\diamond (\mathit{whom} \circ_\diamond (\mathit{Ed} \circ_\diamond \mathit{saw}))) \vdash \mathsf{n}}[\backslash_\diamond E]$$

The crucial step for the extraction is where the structural rule RA applies and puts the assumption on the periphery so that it may be released by slash introduction — before that, it is buried in an inaccessible position. The parallel with traces and movement operations in mainstream generative grammar should be clear from this example (see section 5 for pointers to work on such connections). However, it should be stressed that this is *not* actual movement, but reasoning about syntactic types in a structured set of premises.

This sort of associative restructuring also allows the system to handle right-node raising: *Ed saw* and *Ned heard* both have the type s/np (as they did in CCG derivation (54)). If the categories of coordinators such as *and* are of the form $(\mathsf{X}\backslash_\star\mathsf{X})/_\star\mathsf{X}$, then *Ed saw and Ned heard* has the type s/np as desired.

The set of structural rules can be expanded to permit other ways to reconfigure structured antecedents. For example, the following rules support reordering:

(92) *Left Permutation*: $$\dfrac{(\Delta_a \circ_i (\Delta_b \circ_j \Delta_c)) \vdash \mathsf{X}}{(\Delta_b \circ_j (\Delta_a \circ_i \Delta_c)) \vdash \mathsf{X}}[LP] \qquad (\text{for } i,j \in \{\times,\cdot\})$$

(93) *Right Permutation*: $$\dfrac{((\Delta_a \circ_i \Delta_b) \circ_j \Delta_c) \vdash \mathsf{X}}{((\Delta_a \circ_j \Delta_c) \circ_i \Delta_b) \vdash \mathsf{X}}[RP] \qquad (\text{for } i,j \in \{\times,\cdot\})$$

Consider heavy-NP shift examples such as *Ed saw today his tall friend Ted*. The proof would proceed as it did for *saw Ted today* in (86), and then invoke right permutation:

(94)
$$\dfrac{\mathit{Ed} \vdash \mathsf{np} \qquad \dfrac{\vdots}{((\mathit{saw} \circ \mathit{his\text{-}tall\text{-}friend\text{-}Ted}) \circ \mathit{today}) \vdash \mathsf{s}\backslash_\diamond\mathsf{np}}[RP]}{(\mathit{Ed} \circ_\diamond ((\mathit{saw} \circ \mathit{his\text{-}tall\text{-}friend\text{-}Ted}) \circ \mathit{today})) \vdash \mathsf{s}}[\backslash_\diamond E]$$

Similar use of the permutation rules would allow a number of other permutation possibilities, such as those needed for non-peripheral extraction and scrambling. Of course, there are constructions which do not permit such freedom, and it is here that the multimodal system shines since it allows *selective* access to the structural rules. What this means is that the types of slashes specified on specific lexical entries will interact with the universal grammar (base

logic plus structural rules) to obtain the right behavior. Notice that the $\star$ modality is not referenced in the associative and permutative restructuring rules: it is thus limited to the base logic and thereby forces strict non-associativity and non-permutativity, as it did with the CCG examples in the previous section. See Moot (2002a) for proofs connecting different types of structural rules to the generative capacity they engender.

With the base logic and the structural rules given above, many schematic rules that are commonly employed in rule-based approaches can be shown to be theorems, similarly to the above proof for type-raising. For example, the following is a proof of the backward crossed composition rule of CCG.

$$(95)\qquad \dfrac{\dfrac{\dfrac{\dfrac{\Delta \vdash \mathsf{Y}/_{\times}\mathsf{Z} \qquad [z_1 \vdash \mathsf{Z}]^1}{(\Delta \ \circ_{\times}\ z_1) \vdash \mathsf{Y}}\,[/_{\times}E] \qquad \Gamma \vdash \mathsf{X}\backslash_{\times}\mathsf{Y}}{((\Delta \ \circ_{\times}\ z_1) \ \circ_{\times}\ \Gamma) \vdash \mathsf{X}}\,[\backslash_{\times}E]}{((\Delta \ \circ_{\times}\ \Gamma) \ \circ_{\times}\ z_1) \vdash \mathsf{X}}\,[RP]}{(\Delta \ \circ_{\times}\ \Gamma) \vdash \mathsf{X}/_{\times}\mathsf{Z}}\,[/_{\times}I]^1$$

This proof shows that the rule $\mathsf{Y}/_{\times}\mathsf{Z}\ \mathsf{X}\backslash_{\times}\mathsf{Y} \Rightarrow \mathsf{X}/_{\times}\mathsf{Z}$ is valid given the base logic and the structural rule of Right Permutation. The fact that such rules can be shown as theorems of the base logic plus a set of structural rules provides a crucial connection to CCG and the use of modalities there to control applicability of its finite set of combinatory rules (Baldridge and Kruijff, 2003). This issue is discussed further in the next section.

Researchers in CTL consider a much wider range of logical operators than the binary ones considered here. Most commonly used are the unary connectives, which are used for both features and fine-grained structural control. In fact, some eschew the sort of multimodal binary system given above in favor of unary modalities which govern the applicability of structural rules (e.g., Vermaat (2005)). Bernardi and Szabolcsi (2008) is a detailed syntactic study of quantifier scope and negative polarity licensing in Hungarian that makes extensive use of unary operators, including Galois connectives, which were introduced by Areces and Bernardi (2004) and developed further by Areces *et al.* (2004). Bernardi and Moortgat (2010) discuss recent directions in CTL, including Galois connectives and the Lambek-Grishin calculus. Another recent, related line of work is that of pregroup grammars (Casadio and Lambek, 2008).

4 Relating type-logical and combinatory approaches

By the late 1960's, interest in the AB calculus had been greatly reduced due to the perception that it was basically context-free grammar in different clothing. This was fair to the extent that it had the same generative capacity as context-free grammar (Bar-Hillel *et al.*, 1964), but it was unfair in that it provided a very different architecture for framing theories of grammar and very different means of extending it. However, the success of Montague grammar in the 1970's and its close connection with rule-based extensions of categorial grammar led to a

revival of interest in the syntactic potential of categorial grammars. This in turn spurred a renewed interest in the Lambek calculus in the 1980's, culminating in the type-logical approach introduced in the previous section.

Development in the logical and combinatory traditions proceeded largely independently of one another until the early 2000's. The former was largely concerned with linguistic applicability of different logical operators and proof-theoretic properties of the proposed logical systems. The latter focused more on obtaining linguistically expressive systems with low automata-theoretic power and attractive computational properties. The logically-minded categorial grammarian will complain that the combinatory approaches are incomplete, partial systems of type-based inference (and therefore of less interest). The combinatory-minded one will complain that the logical approaches are impractical for computational applications (and therefore of less interest).

Despite these historical differences, it is important to realize that the actual *linguistic* ramifications of both approaches are largely compatible. Furthermore, we can generate rule-based systems from an underlying logic (Baldridge and Kruijff, 2003; Hoyt and Baldridge, 2008); from this perspective, we can investigate and develop the underlying logic while enjoying the computational advantages of a finite rule set whose rules ensure that each derivational step leads to a reduction, rather than expansion, of categories.

As noted earlier, Jacobson (1992b) used slash-types in a rule-based categorial grammar that made it possible to force some categories to compose but not to apply. This is an interesting alternative that is available to a multimodal rule-based system. However, from the CTL perspective *all* slashes can be eliminated. To achieve this effect in a rule-based system derived from a CTL system, if desired, would require appropriate use of unary modalities (the lock-and-key strategy (Moortgat, 1997)).

With a CCG rule set derived from a CTL system, we obtain the same basic analyses as we would with the original system. In one sense, the CCG derivations can be seen as abbreviated versions of the corresponding CTL proofs. Hypothetical reasoning and explicit structural reconfiguration do not play a role: instead, they are folded into the composition and type-raising rules. Of course, since CTL systems are not finitely axiomatizable, it would offer further derivational possibilities than the finite CCG rule set.

The claims about the grammars (and the possible analyses they support) are for most purposes identical given a CTL system defined with the structural rules suggested in the previous section and CCG standard rule set. Regardless of the categorial framework, the linguistic work is done in the categories—they are where the bulk of the linguistic claims are made. This is to say that despite many apparent surface differences, logical and rule-based categorial systems are mostly compatible with respect to their treatment of syntactic phenomena. Viewed in this way, CTL provides a microscope that allows us to peer into very low level details concerning properties such as associativity and commutativity; it then tells us how we can prove rule schemata that are part of the CCG rule base, or which have a similar nature to CCG rules. With those rules in hand, we can essentially short-cut many of the logical steps needed to complete

certain CTL inferences. This means not only shorter derivations, but also many advantages for practical parsing with categorial grammars.

Generative capacity: The perceived position of natural language on the Chomsky hierarchy (for those who believe it has such a position) has fluctuated during the past five decades. During the the 1960's and 1970's, it was generally accepted by linguists that grammars for natural languages required greater than context-free power. Due to this belief, AB categorial grammar was somewhat sidelined after it was proved to be context-free by Bar-Hillel, Gaifman, and Shamir in 1964 — even Bar-Hillel himself gave up on categorial grammar because of this result. Bar-Hillel should not have despaired so easily — both the belief that categorial grammar was just a notational variant of context-free grammar and the apparent supposition that the categorial approach could not somehow be generalized to create systems of greater power were incorrect.

Context-free grammars are indeed now known to be not even *weakly* adequate to handle all natural language constructions, such as crossing dependencies in Swiss German (Huybregts, 1984; Shieber, 1985). The combinatory rules employed by CCG increase the context-free power of AB by a small but linguistically significant amount. This places CCG into the class of mildly context-sensitive formalisms (Vijay-Shanker and Weir, 1994), which are able to capture crossing dependencies, both with respect to the string languages that characterize them (weak generative capacity) and the structural descriptions which capture the appropriate dependencies (strong generative capacity). As a mildly context-sensitive formalism, CCG enjoys worst-case polynomial parsing complexity (n^6) (Vijay-Shanker and Weir, 1990).

Categorial systems with more power have been defined. Motivated by scrambling phenomena in languages such as Turkish, Multiset-CCG modifies the manner in which categories are constructed and defines rules of combination for them which allow greater flexibility, resulting in a formalism with more than mildly context-sensitive power (Hoffman, 1995). CTL using very general structural rules of permutation is Turing complete (Carpenter, 1995). Moot (2002b) provides a much finer characterization of the different generative strengths produced by CTL systems which conform to certain constraints, including the result that systems which use non-expanding structural rules are only context-sensitive. This result is particularly interesting since it appears that nearly all linguistic applications of CTL have obeyed this constraint (Moot, 2002b).

After a long period of relative dormancy as regards research into generative power, there has been a renewed interest in the *strong* generative capacity of CCG (and Tree-Adjoining Grammar). The upshot of this work is that the equivalences are not so tidy in this regard (Hockenmaier and Young, 2008; Koller and Kuhlmann, 2009). Furthermore, even the weak generative capacity can be greatly affected by subtle choices, such as the use of rule restrictions or a particular set of modalities in CCG (Kuhlmann *et al.*, 2010).

The issue of generative capacity was a historical sticking point between the CCG and CTL traditions, though it is not a central concern now. Early attempts to extend the Lambek calculus to allow for permutation led to full permutation collapse (Moortgat, 1988). CTL, however, does not suffer from such a collapse

since commutative operations can be introduced in a controlled fashion with modal control. Nonetheless, generative capacity does still underly a difference of explanatory philosophy between linguistic applications of the two frameworks. For CTL researchers, issues of generative capacity are generally not considered to be of prime theoretical importance. In contrast, work in both CCG and its closest sibling formalism, Tree Adjoining Grammar (TAG) (Joshi, 1988), takes a committed stance on minimizing generative power. The basic linguistic claim for such formalisms is that their restricted formal power provides inherent limitations on theories which are based on them (e.g., see Frank (2002)). In this way, they enforce a wide range of formal universals. Such formalisms sit on the lower bound of natural language complexity without venturing any further — they are expressive enough, but cannot do everything.

With respect to generative capacity, a key attraction of the multimodal approach is that it is able to mix systems of varying power in a resource-sensitive manner. Thus, more powerful operations of grammatical combination—should their inclusion be warranted by linguistic evidence—are introduced in a *controlled* manner.

5 Related formalisms

Categorial grammars of all kinds share deep connections to many other approaches to natural language grammar.

Tree Adjoining Grammar (Joshi, 1988) is much like CCG in that it is also highly lexicalized and assumes a small, universal rule component. As mentioned above, it is also mildly context-sensitive, and like CCG it has been the focus of a great deal of computational work. In general, there has been a great deal of intellectual crossover between CCG and TAG.

Categorial grammar's extreme lexicalism ties it closely to the tradition of dependency grammar (Hudson, 1984; Sgall *et al.*, 1986), which also focuses on the way in which patterns of semantic linkage hold a sentence together, rather than segmenting sentences according to analytic patterns such as phrase structure. The use of typed-feature structures in CCG and related rule-based CGs (Zeevat *et al.*, 1987; Villavicencio, 2002; Baldridge, 2002; Beavers, 2004; McConville, 2006) is informed by much work in Head-driven Phrase Structure Grammar (Pollard and Sag, 1994; Sag *et al.*, 2003) and its predecessors. This is especially true with respect to providing a theory of the lexicon. In this exposition, we have provided a flat lexicon where no information is shared between the categories. While useful for explicating how categorial grammar works at the derivational level, it is clearly an unrealistic way to organize what is, after all, the core of grammar. There are a number of solutions to manage redundancy and category relatedness in a lexicon. Using typed-feature structures with inheritance as is common in HPSG, it is possible to define a structured lexicon that eliminates a great deal of redundancy between categories of different arity. For example, the ditransitive category can be based on the transitive category, which in turn can be based on the intransitive category, and so on (Villavicen-

cio, 2002; Baldridge, 2002; McConville, 2006). Another related strategy is to use lexical rules that produce additional categories based on core lexical category assignments (Carpenter, 1992); such rules are similar to those used in Generalized Phrase Structure Grammar (Gazdar *et al.*, 1985) and Lexical Functional Grammar (Kaplan and Bresnan, 1982).

There are other strong connections with HPSG and LFG. HPSG analyses typically rely on a very abstract set of phrase-structure rules that are similar to the rule schemata used in CCG. As with categorial grammar, HPSG lexical items trigger derivational steps via their subcategorization requirements. Interestingly, Morrill (2007) comments that HPSG is in fact a variety of categorial grammar: this is hardly a stretch given Pollard's type-logical formulation of HPSG (Pollard, 2004). His recent Convergent Grammar (Pollard, 2008) synthesizes ideas from both categorial grammar and HPSG, and is closely related to Abstract Categorial Grammars (de Groote, 2001) and λ-Grammar (Muskens, 2007).

Many researchers working in constraint-based approaches, especially HPSG, have adopted construction grammar (Goldberg, 2006) as an organizing philosophy. It might seem that construction grammar, with its emphasis on conventionalized and non-compositional grammatical processes, would be incompatible with categorial grammar. However, there seems to be every reason to use categorial grammar as a formalism for analyses compatible with construction grammar. As an example in this direction, Steedman and Baldridge (2011) discuss the "way" construction as in *Marcel slept his way to the top.* They suggest that one of the categories for *his* to license this construction would essentially incorporate *way* as an item that is subcategorized for.

(96) $\mathbf{his} \vdash ((\mathsf{s}\backslash\mathsf{np})\backslash_{\mathsf{LEX}}(\mathsf{s}\backslash\mathsf{np}))/\mathsf{pp}/''\mathsf{way}'' : \lambda i\lambda p\lambda q\lambda y.cause'(iterate'(qy))(result'(py))$

This strategy, which is related to the use of trees with multiple lexical anchors in TAG, can be used to lexicalize many other phenomena identified by construction grammarians. If pursued seriously, it would undoubtedly involve an examination of the theory of the categorial lexicon and ways of managing the consequent proliferation of category types. Arguably, this is where the focus of attention of linguistic work in categorial grammar should be anyway.

Johnson (1999) provides a resource-sensitive interpretation of LFG that builds on many ideas from CTL. See Muskens (2001) for further discussion of the relationship between categorial grammar and LFG and a proposal for providing a hybrid of the two approaches, building on ideas from Oehrle (1999).

The basic architectures of theories of grammar based on both CTL and CCG (mostly unchanged since the late 1980's) are largely in accord with many of the principles later advocated in some versions of the Minimalist program (Chomsky, 1995). A deeper and perhaps surprising (to some) connection appears when Minimalism is viewed through the lens of Minimalist grammars (Stabler, 1997; Cornell, 1999). As noted earlier, hypothetical reasoning in CTL combined with structural reasoning has many parallels with movement operations as construed in Minimalism. For several perspectives on the relation between CTL and Minimalism, see the collection of papers in the journal of *Research on Logic*

and Computation, 2004, Volume 2(1) (in particular, Retoré and Stabler (2004), Lecomte (2004), Vermaat (2004) and Cornell (2004)) and Vermaat (2005).

6 Computational applications

Along with much progress with respect to formalizing varieties of categorial grammar with desirable linguistic properties, there has also been considerable development in computational applications of categorial grammar. The success of categorial grammar in these arenas is in great part based on its high degree of lexicalization and its semantic transparency.

Like many other computationally amenable frameworks, there are grammar development environments available for testing analyses. The Grail system allows CTL structural rule packages and lexicons to be defined and tested (Moot, 2002b). The OpenCCG system similarly supports (multi-modal) CCG grammar development and performs both sentence parsing and realization; it has also been used for a wide-range of dialog systems—see Baldridge *et al.* (2007) for discussion of OpenCCG grammar development and applications and White (2006) for specific discussion of efficient realization with OpenCCG.

A major development was the creation of CCGbank (Hockenmaier and Steedman, 2007), which has allowed the creation of fast and accurate statistical CCG parsers for producing deep dependencies (Hockenmaier, 2003; Clark and Curran, 2007). A key feature of categorial grammar, the fact that lexical categories are highly informative for overall syntactical analysis, is used in the C&C CCG parser of Clark and Curran (2007) to make it among the fastest of wide-coverage statistical parsers that produce deep dependencies. A fast *supertagger* is used to perform assignment of lexical categories before parsing begins, thereby drastically reducing the structural ambiguity that must be considered during parsing. This adaptive supertagging strategy is exploited by Kummerfeld *et al.* (2010) in combination with self-training to achieve fast parsing times with no loss in accuracy. Auli and Lopez (2011b) is a related study that takes a different strategy for reducing the potential for cutoffs to reduce accuracy: they explore parsing CCGs with adaptive supertagging techniques and A* search to both explore the trade-offs made by supertag cutoffs and obtain faster parsing times. In another paper, the same authors investigate an alternative model that integrates the features of a supertagger and parser into the same model (Auli and Lopez, 2011a). The best performance to date on CCG parsing for CCGbank is obtained by this model optimized for F-measure (Auli and Lopez, 2011c).

CCGbank has provided a basis for applications other than parsing. For example, supertaggers learned from CCGbank have been used to improve statistical machine translation systems (Birch *et al.*, 2007; Hassan *et al.*, 2007), and discriminative models over CCG derivations have been used for word-ordering (generating a sentence from a multi-set of input words) (Zhang *et al.*, 2012). Analyses from parsers have been used for semantic role labeling (Gildea and Hockenmaier, 2003; Boxwell *et al.*, 2011). OpenCCG grammars that support wide-coverage sentence realization have been bootstrapped from CCGbank, re-

ducing the effort that goes into creating larger grammars while taking advantage of the deep representations supported by OpenCCG (Espinosa *et al.*, 2008).

A number of augmentations of CCGbank have been created, such as improving the representation of noun-phrases (Vadas and Curran, 2008), fully lexicalizing type-changing rules (using hat categories) (Honnibal and Curran, 2009), and adding verb-particle constructions (Constable and Curran, 2009). Many of these augmentations were integrated in the *rebanking* of CCGbank completed by Honnibal *et al.* (2010). Most recently, the resources—both data and processing tools—that have been built up around CCGbank are being used to bootstrap broader and deeper annotations for the Groningen Meaning Bank, using an ongoing, collaborative, semi-automated annotation environment (Basile *et al.*, 2012).

Unlike CCG, CTL has seen little use in computational applications. This is in large part due to significant challenges in efficiently dealing with the options made available by using the full logic, which typically allows many more ways to bracket a string than CCG's (structurally incomplete) finite rule set permits. For recent work in parsing restricted CTL systems, see Capelleti (2007) and Fowler (2008). Interestingly, Capelleti also considers parsing a variant of CCG with the product operator. This sort of strategy seems to be the most expedient way to efficiently parse CTL systems: basically, one could compile CCG-like rules on an as-needed basis and then use standard parsing algorithms that have been successfully used with CCG.

Of course, the work mentioned above assumes that we have defined a grammar manually, either explicitly in a grammar development environment or implicitly in the derivations of sentences in a corpus. It is naturally an interesting question whether we can learn categorial grammars from less informative starting points. Villavicencio (2002) learns lexicons for a rule-based CG given child-directed speech annotated with logical forms. There are recent efforts in this direction that induce CCG parsers from sentences paired with logical forms (Zettlemoyer and Collins, 2007; Kwiatkowski *et al.*, 2011). There, a small set of category schemata and paired, abstract logical forms are assumed, and the mapping from words to the appropriate categories and lexical semantics is then learned. Other work has considered how to extend an initial seed lexicon using grammar-informed Hidden Markov Models (Baldridge, 2008; Ravi *et al.*, 2010) and inference from a partially completed parse chart (Thomforde and Steedman, 2011).

7 Conclusion

Categorial grammar has a long, but punctuated, history that has coalesced in the last thirty years to provide unique perspectives on natural language grammar. There are connections not only to combinatory logic and resource-sensitive linear logic, but also category theory more generally. The controlled associativity and permutativity available to modern categorial grammars allows them to enjoy straightforward analyses for a number of otherwise troubling issues in

constituency without being overly permissive. Work in Categorial Type Logics continues to explore new type constructors that may have linguistic application and explore their logical/mathematical properties. Work in Combinatory Categorial Grammar remains focused on practical applications and grammar acquisition, both from annotated resources such as CCGbank and from text alone using machine learning methods. Given the connections between the two traditions, briefly sketched out here, it is now easier to translate innovations from one to the other. With the current set of formal and computational advances and their application to language phenomena, categorial grammar is well placed to further deepen our understanding of natural language grammar, both through standard linguistic investigation into specific languages and constructions and as the basis for acquisition of grammars using machine learning methods on text or in communicative agents such as chatbots and robots.

8 Further reading

There is an incredibly diverse number of systems based on or related to categorial grammar, and a bewildering (and almost equally diverse) set of notations for those systems. Due to space limitations, much of this work has been only briefly touched on: the interested reader can hopefully remedy some of these gaps by looking further into some of the following pointers.

See Steedman's article on categorial grammar in the previous edition of this handbook for pointers to further reading prior to 1993. That article covers many issues and topics that the present one does not, such as more detailed consideration of combinators, the relation of categorial grammar to linear-indexed grammars, treatments of quantification, and responses to common criticism of categorial grammar at the time.

Published around the same time, Wood (1993) provides a very complete and balanced introduction to many ideas, analyses and issues in categorial grammar. The collection of papers in Kruijff and Oehrle (2003) are recent contributions within both the combinatory and type-logical traditions. Morrill (2007) gives a literature survey covering major developments in categorial grammar from 1935 to 1994, with a particular emphasis on type-logical work.

Morrill's most recent book, *Categorial Grammar: Logical Syntax, Semantics, and Processing* (Morrill, 2011), is an up-to-date introduction and extended exposition on the logical tradition in categorial grammar. In particular, it develops discontinuous Lambek grammars, an extension of type-logical grammars that support a new analysis of gapping, following Hendriks (1995), and it covers processing and parsing from the type-logical perspective, building on ideas in Morrill (2000). Some earlier books covering the type-logical approach are Moortgat (1988), Morrill (1994), and Carpenter (1998). A notable aspect of Morrill's book is the thorough discussion of Montague grammar and how it can be formulated within a modern categorial grammar framework.

There are also several notable article-length discussions of the type-logical approach. Moortgat's 2010 chapter in the *Handbook of Logic and Language*

provides an up-to-date and definitive overview of CTL (updating the previous version, Moortgat (1997), in the first edition of that handbook). Oerhle's article in Borsley and Börjars (2011) provides a particularly readable introduction to CTL. It is worth noting that much of the literature on CTL is highly technical and difficult going for newcomers. In this regard, Barker (2003) is an excellent and friendly source for understanding many fundamental concepts in CTL, and Vermaat (2005) gives one of the most accessible introductions to CTL as a whole.

A fairly recent development in the space of type-logical approaches is Lambek's pregroup grammar (Lambek, 1999), a type-based algebraic approach that seeks to use standard mathematical structures and concepts for modeling natural language grammar. See Lambek (2000, 2007, 2010) and the collection of papers in Casadio and Lambek (2008) for articles on the formal and linguistic properties of pregroup grammars.

Pregroup grammars have also been used to provide a solution to a recent problem in natural language semantics, in which distributional models of lexical semantics—based on vector spaces induced from large corpora—have been combined with traditional compositional methods. Coecke *et al.* (2011) show how a type-driven semantic compositional model can be developed in the vector-space setting, in which the syntactic types of the pregroups correspond to tensor product spaces. Vectors for relational words, such as verbs and adjectives, live in the appropriate tensor product space. The framework also provides a semantic operation analagous to function application in which a verb vector, for example, can be "applied" to its arguments, resulting in a vector living in a space for sentences. The ultimate goal of such efforts is to devise systems that are able to assign detailed (possibly hierarchical), data-driven meaning representations to sentences. In these representations, the meaning of *life* is not (the constant) *life'*: instead, words and phrases have rich internal representations that allow one to calculate their similarity with other words and phrases. Such similarity comparisons can then be used for other computations such as making inferences based on the substitutability of different predications with respect to one another. See Erk (2010) for an overview of much current work in this vein.

Steedman's *Surface Structure and Interpretation* monograph provides a detailed account of core theoretical linguistic concerns from the perspective of Combinatory Categorial Grammar, with a special emphasis on extraction asymmetries, coordination, and their interaction with binding (Steedman, 1996a). His later book *The Syntactic Process* (Steedman, 2000b) is a thorough exposition on both formal and theoretical aspects of Combinatory Categorial Grammar that encapsulates the content of many of the papers written on CCG through the 1980's and 1990's. The article by Steedman and Baldridge in Borsley and Börjars (2011) gives a current and detailed account of CCG and analyses of a wide-range of bounded and unbounded language phenomena. Finally, Steedman's latest book, *Taking Scope* (Steedman, 2012), develops a surface-compositional account of of quantification using CCG. In doing so, it covers a great deal of linguistic ground and touches on both computational and human sentence processing.

References

Ades, Anthony and Steedman, Mark, 1982. "On the order of words." *Linguistics & Philosophy* 7:639–642.

Ajdukiewicz, Kazimierz, 1935. "Die syntaktische Konnexität." In Storrs McCall (ed.), *Polish Logic 1920-1939*, Oxford University Press, 207-231. translated from *Studia Philosophica*, 1, 1-27.

Aoun, Joseph and Benmamoun, Elabbas, 1998. "Minimality, Reconstruction, and PF Movement." *Linguistic Inquiry* 29(4):59–597.

Aoun, Joseph and Choueiri, Lena, 2000. "Epithets." *Natural Language and Linguistic Theory* 18:1–39.

Aoun, Joseph, Choueiri, Lina, and Hornstein, Norbert, 2001. "Resumption, Movement, and Derivational Economy." *Linguistic Inquiry* 32(3):371–403.

Aoun, Joseph and Li, Audrey, 2003. *Essays on the Representational and Derivational Nature of Grammar*. MIT Press (Cambridge).

Areces, C. and Bernardi, R., 2004. "Analyzing the Core of Categorial Grammar." *Journal of Logic, Language and Information* 13(2):121–137.

Areces, C., Bernardi, R., and Moortgat, M., 2004. "Galois connections in categorial type logic." In R. Oehrle and L. Moss (eds.), *Electronic Notes in Theoretical Computer Science. Proceedings of FGMOL'01*. Elsevier Science B.V., volume 53, 1–12.

Asudeh, Ash, 2005. "Relational Nouns, Pronouns, and Resumption." *Linguistics and Philosophy* 28:375–446.

Asudeh, Ash, 2012. *The Logic of Pronominal Resumption*. Oxford University Press.

Auli, Michael and Lopez, Adam, 2011a. "A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing." In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, 470–480.

Auli, Michael and Lopez, Adam, 2011b. "Efficient CCG Parsing: A* versus Adaptive Supertagging." In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, 1577–1585.

Auli, Michael and Lopez, Adam, 2011c. "Training a Log-Linear Parser with Loss Functions via Softmax-Margin." In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, 333–343.

Bach, Emmon, 1979. "Control in Montague Grammar." *Linguistic Inquiry* 10:513–531.

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

Baldridge, Jason, 2008. "Weakly Supervised Supertagging with Grammar-Informed Initialization." In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Manchester, UK: Coling 2008 Organizing Committee, 57–64.

Baldridge, Jason, Chatterjee, Sudipta, Palmer, Alexis, and Wing, Ben, 2007. "DotCCG and VisCCG: Wiki and Programming Paradigms for Improved Grammar Engineering with OpenCCG." In *Proceedings of the Workshop on Grammar Engineering Across Frameworks*. Stanford, CA.

Baldridge, Jason and Kruijff, Geert-Jan, 2002. "Coupling CCG and Hybrid Logic Dependency Semantics." In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, 319–326.

Baldridge, Jason and Kruijff, Geert-Jan, 2003. "Multi-Modal Combinatory Categorial Grammar." In *Proceedings of EACL*. Budapest, Hungary.

Bar-Hillel, Yehoshua, 1953. "A Quasi-arithmetical Notation for Syntactic Description." *Language* 29:47–58.

Bar-Hillel, Yehoshua, Gaifman, C., and Shamir, E., 1964. "On categorial and phrase structure grammars." In Yehoshua Bar-Hillel (ed.), *Language and information*, Reading MA: Addison-Wesley. 99–115. 1964.

Barker, Chris, 2003. "A gentle introduction to Type Logical Grammar, the Curry-Howard correspondence, and cut-elimination." semanticsarchive.net.

Basile, Valerio, Bos, Johan, Evang, Kilian, and Venhuizen, Noortje, 2012. "A platform for collaborative semantic annotation." In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Avignon, France.

Beavers, John, 2004. "Type-inheritance Combinatory Categorial Grammar." In *Proceedings of COLING-04*. Geneva, Switzerland.

Bernardi, Raffaella and Moortgat, Michael, 2010. "Continuation semantics for the Lambek-Grishin calculus." *Information and Computation* 208(5):397–416.

Bernardi, Raffaella and Szabolcsi, Anna, 2008. "Optionality, scope, and licensing: An application of partially ordered categories." *Journal of Logic, Language and Information* 17(3):237–283.

Birch, Alexandra, Osborne, Miles, and Koehn, Philipp, 2007. "CCG Supertags in Factored Translation Models." In *Proceedings of the Second Workshop on Statistical Machine Translation*. Prague, 9–16.

Borsley, Robert and Börjars, Kersti (eds.), 2011. *Non-Transformational Syntax: Formal and Explicit Models of Grammar*. New York: Wiley-Blackwell.

Boxwell, Stephen, Brew, Chris, Baldridge, Jason, Mehay, Dennis, and Ravi, Sujith, 2011. "Semantic Role Labeling Without Treebanks?" In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Chiang Mai, Thailand: Asian Federation of Natural Language Processing, 192–200.

Capelleti, Matteo, 2007. *Parsing with Structure-Preserving Categorial Grammars*. Ph.D. thesis, Utrecht.

Carpenter, Bob, 1992. "Lexical and Unary Rules in Categorial Grammar." In Robert Levine (ed.), *Formal Grammar: Theory and Implementation*, Oxford University Press, volume 2 of *Vancouver Studies in Cognitive Science*.

Carpenter, Bob, 1995. "The Turing-Completeness of Multimodal Categorial Grammar." ms, http://www.illc.uva.nl/j50/contribs/carpenter/index.html.

Carpenter, Bob, 1998. *Type-Logical Semantics*. Cambridge Massachusetts: The MIT Press.

Casadio, Claudia, 1988. "Semantic Categories and the Development of Categorial Grammars." In Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler (eds.), *Categorial Grammars and Natural Language Structures*, Dordrecht: Reidel. 95–123. Proceedings of the Conference on Categorial Grammar,Tucson, AR, June 1985.

Casadio, Claudia and Lambek, Joachim (eds.), 2008. *Computational Algebraic Approaches to Natural Language*. Polimetrica.

Chomsky, Noam, 1995. *The Minimalist Program*. Cambridge, Mass.: MIT Press.

Choueiri, Lena, 2002. *Issues in the Syntax of Resumption: Restrictive Relatives in Lebanese Arabic*. Ph.D. thesis, University of Southern California.

Clark, Stephen and Curran, James, 2007. "Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models." *Computational Linguistics* 33(4).

Coecke, Bob, Sadrzadeh, Mehrnoosh, and Clark, Stephen, 2011. "Mathematical Foundations for a Compositional Distributional Model of Meaning." *Linguistic Analysis 36: A Festschrift for Joachim (Jim) Lambek* .

Constable, James and Curran, James, 2009. "Integrating Verb-Particle Constructions into CCG Parsing." In *Proceedings of the Australasian Language Technology Association Workshop 2009*. Sydney, Australia, 114–118.

Copestake, Ann, Lascarides, Alex, and Flickinger, Dan, 2001. "An Algebra for Semantic Construction in Constraint-based Grammars." In *Proceedings of the 39th Annual Meeting of the Association of Computational Linguistics*. Toulouse, France, 132–139.

Cornell, Thomas, 1999. "Representational Minimalism." In Hans-Peter Kolb and Uwe Mönnich (eds.), *The Mathematics of Syntactic Structure: Trees and Their Logics*, Walter de Gruyter. 301–340.

Cornell, Thomas, 2004. "Lambek Calculus for Transformational Grammar." *Research on Language and Computation* 2(1):105–126.

Curry, Haskell B. and Feys, Robert, 1958. *Combinatory Logic: Vol I.* North Holland, Amsterdam.

Demirdache, Hamida, 1991. *Resumptive Chains in Restrictive Relatives, Appositives and Dislocation Structures*. Ph.D. thesis, MIT.

Demirdache, Hamida, 1997. "Dislocation, Resumption, and Weakest Crossover." In Elena Anagnostopoulou, Henk Van Riemsdijk, and Frans Zwarts (eds.), *Materials on Left-Dislocation*, John Benjamins (Philadelphia). 193–231.

Dowty, David, 1982. "Grammatical Relations and Montague Grammar." In P. Jacobson and G. K. Pullum (eds.), *The Nature of Syntactic Representation*, Dordrecht: Reidel. 79–130.

Dowty, David, 1988. "Type-raising, Functional Composition, and Non-constituent Coordination." In Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler (eds.), *Categorial Grammars and Natural Language Structures*, Dordrecht: Reidel. 153–198. Proceedings of the Conference on Categorial Grammar,Tucson, AR, June 1985.

Dowty, David, 1997. "Non-Constituent Coordination, Wrapping,and Multimodal Categorial Grammars." In M. L. Dalla Chiara et al. (ed.), *Structures and Norms in Science*, Dordrecht: Kluwer. 347–368.

Erk, Katrin, 2010. "What Is Word Meaning, Really? (And How Can Distributional Models Help Us Describe It?)." In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*. Uppsala, Sweden: Association for Computational Linguistics, 17–26.

Espinosa, Dominic, White, Michael, and Mehay, Dennis, 2008. "Hypertagging: Supertagging for Surface Realization with CCG." In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*. Columbus, OH, 183–191.

Fowler, Timothy A. D., 2008. "Efficiently Parsing with the Product-Free Lambek Calculus." In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Manchester, UK: Coling 2008 Organizing Committee, 217–224.

Frank, Robert, 2002. *Phrase Structure Composition and Syntactic Dependencies*. Cambridge, MA: MIT Press.

Gazdar, Gerald, Klein, Ewan, Pullum, Geoffrey K., and Sag, Ivan A., 1985. *Generalised Phrase Structure Grammar*. Oxford: Blackwell.

Gildea, Daniel and Hockenmaier, Julia, 2003. "Identifying Semantic Roles Using Combinatory Categorial Grammar." In Michael Collins and Mark Steedman (eds.), *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*. 57–64.

Girard, Jean-Yves, 1987. "Linear Logic." *Theoretical Computer Science* 50:1–102.

Goldberg, Adele, 2006. *Constructions at Work*. Oxford University Press.

de Groote, Philippe, 2001. "Towards Abstract Categorial Grammars." In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*. Toulouse, France: Association for Computational Linguistics, 252–259.

Hassan, Hany, Sima'an, Khalil, and Way, Andy, 2007. "Supertagged Phrase-Based Statistical Machine Translation." In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. Prague, 288–295.

Hendriks, Petra, 1995. *Comparatives and Categorial Grammar*. Ph.D. thesis, Rijksuniversiteit Groningen.

Hepple, Mark, 1990. *The Grammar and Processing of Order and Dependency: A Categorial Approach*. Phd, University of Edinburgh.

Hepple, Mark, 1995. "Hybrid Categorial Logics." *Bulletin of the IGPL* 3(2,3):343–356. Special Issue on Deduction and Language.

Hockenmaier, Julia, 2003. "Parsing with Generative Models of Predicate-Argument Structure." In *Proceedings of ACL*.

Hockenmaier, Julia and Steedman, Mark, 2007. "CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank." *Computational Linguistics* 33(3):355–396.

Hockenmaier, Julia and Young, Peter, 2008. "Non-local scrambling: the equivalence of TAG and CCG revisited." In *Proceedings of The Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+9)*. Tbingen, Germany.

Hoffman, Beryl, 1995. *Computational Analysis of the Syntax and Interpretation of 'Free' Word-order in Turkish*. Ph.D. thesis, University of Pennsylvania. IRCS Report 95-17.

Honnibal, Matthew, 2009. *Hat Categories: Representing Form and Function Simultaneously in Combinatory Categorial Grammar*. Ph.D. thesis, University of Sydney.

Honnibal, Matthew and Curran, James R., 2009. "Fully Lexicalising CCGbank with Hat Categories." In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Honnibal, Matthew, Curran, James R., and Bos, Johan, 2010. "Rebanking CCGbank for Improved NP Interpretation." In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, 207–215.

Hoyt, Frederick and Baldridge, Jason, 2008. "A Logical Basis for the D combinator and normal form constraints in Combinatory Categorial Grammar." In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. Columbus, 326–334.

Hoyt, Frederick M, 2010. *Negative Concord in Levantine Arabic*. Ph.D. thesis, University of Texas at Austin.

Hudson, Richard, 1984. *Word Grammar*. Oxford: Blackwell.

Huybregts, Riny, 1984. "The Weak Inadequacy of Context-free Phrase-structure Grammars." In Ger de Haan, Mieke Trommelen, and Wim Zonneveld (eds.), *Van Periferie naar Kern*, Foris Dordrecht.

Jacobson, Pauline, 1990. "Raising as Function Composition." *Linguistics and Philosophy* 13:423–475.

Jacobson, Pauline, 1992a. "Antecedent-Contained Deletion in Variable-Free Semantics." In Chris Barker and David Dowty (eds.), *Proceedings of the Second Conference on Semantics and Linguistic Theory*. OSU Working Papers in Linguistics, Ohio State University.

Jacobson, Pauline, 1992b. "Flexible Categorial Grammars: Questions and Prospects." In Robert Levine (ed.), *Formal Grammar*, Oxford: Oxford University Press. 129–167.

Jacobson, Pauline, 1993. "i-within-i Effects in a Variable-Free Semantics and a Categorial Syntax." In Paul Dekker and Martin Stokhof (eds.), *Proceedings of the Ninth Amsterdam Colloquium*. 349–369.

Jacobson, Pauline, 1999. "Towards a Variable-Free Semantics." *Linguistics and Philosophy* 22:117–184.

Jacobson, Pauline, 2000. "Paycheck Pronouns, Bach-Peters Sentences, and Variable-Free Semantics." *Natural Language Semantics* 8:77–155.

Jacobson, Pauline, 2002. "Direct Compositionality and Variable-Free Semantics: The Case of Binding into Heads." In Brendan Jackson (ed.), *Proceedings of the Twelfth Conference on Semantics and Linguistic Theory (SALT XII)*.

Jacobson, Pauline, 2003. "Binding without pronouns (and pronouns without binding)." In G.-J. M. Kruijff and R.T. Oehrle (eds.), *Resource-Sensitivity, Binding, and Anaphora*, Kluwer. 57–96.

Jäger, Gerhard, 1996. *Topics in Dynamic Syntax*. Ph.D. thesis, Humboldt-Universität zu Berlin.

Jäger, Gerhard, 1997. "Anaphora and Ellipsis in Type Logical Grammar." In Paul Dekker, Martin Stokhof, and Y Venema (eds.), *Proceedings of the 11th Amsterdam Colloquium*. University of Amsterdam, University of Amsterdam, 175–180.

Jäger, Gerhard, 2001. "Anaphora and quantification in Categorial Grammar." In Michael Moortgat (ed.), *Logical Aspects of Computational Linguistics*, Springer, volume 2014 of *Springer Lecture Notes in Artificial Intelligence*. 70–90.

Jäger, Gerhard, 2005. *Anaphora and Type Logical Grammar*. Springer.

Johnson, Mark, 1999. "A Resource Sensitive Interpretation of Lexical Functional Grammar." *Journal of Logic, Language and Information* 8(1):45–81.

Joshi, Aravind, 1988. "Tree Adjoining Grammars." In David Dowty, Lauri Karttunen, and Arnold Zwicky (eds.), *Natural Language Parsing*, Cambridge: Cambridge University Press. 206–250.

Kaplan, Ronald and Bresnan, Joan, 1982. "Lexical-Functional Grammar: A formal system for grammatical representation." In *The Mental Representation of Grammatical Relations*, Cambridge, MA: MIT Press. 173–281.

Koller, Alexander and Kuhlmann, Marco, 2009. "Dependency trees and the strong generative capacity of CCG." In *Proceedings of the 12th Conference of the European Chapter of the ACL*. Athens.

Kruijff, Geert-Jan M and Oehrle, Richard T. (eds.), 2003. *Resource-Sensitivity, Binding and Anaphora*. Kluwer.

Kuhlmann, Marco, Koller, Alexander, and Satta, Giorgio, 2010. "The importance of rule restrictions in CCG." In *Proceedings of the 48th ACL*. Uppsala.

Kummerfeld, Jonathan K., Roesner, Jessika, Dawborn, Tim, Haggerty, James, Curran, James R., and Clark, Stephen, 2010. "Faster Parsing by Supertagger Adaptation." In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, 345–355.

Kwiatkowski, Tom, Zettlemoyer, Luke, Goldwater, Sharon, and Steedman, Mark, 2011. "Lexical Generalization in CCG Grammar Induction for Semantic Parsing." In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, 1512–1523.

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

Lambek, Joachim, 1999. "Type Grammar Revisited." In *Logical Aspects of Computational Linguistics*, Spring, volume 1582 of *Lecture Notes in Computer Science*. 1–27.

Lambek, Joachim, 2000. "Type grammar meets german word order." *Theoretical Linguistics* (26):19–30.

Lambek, Joachim, 2007. "From word to sentence: a pregroup analysis of the object pronoun who(m)." *Journal of Logic, Language and Information* 16:303–323.

Lambek, Joachim, 2010. "Exploring feature agreement in french with parallel pregroup computations." *Journal of Logic, Language and Information* 19(1):75–88.

Lecomte, Alain, 2004. "Rebuilding MP on a Logical Ground." *Research on Language and Computation* 2(1):27–55.

McConville, Mark, 2006. "Inheritance and the CCG Lexicon." In *Proceedings of the European Association for Computational Linguistics*. Trento, 1–8.

Moortgat, M. and Oehrle, R., 1994. "Adjacency, dependency, and order." In P. Dekker and M. Stokhof (eds.), *Proceedings of the Ninth Amsterdam Colloquium*. Amsterdam: ILLC.

Moortgat, Michael, 1988. *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus*. Dordrecht, The Netherlands: Foris.

Moortgat, Michael, 1997. "Categorial Type Logics." In Johan van Benthem and Alice ter Meulen (eds.), *Handbook of Logic and Language*, Amsterdam New York etc.: Elsevier Science B.V.

Moortgat, Michael, 1999. "Constants of grammatical reasoning." In Gosse Bouma, Erhard W. Hinrichs, Geert-Jan M. Kruijff, and Richard T. Oehrle (eds.), *Constraints and Resources in Natural Language Syntax and Semantics*, Stanford CA: CSLI Publications.

Moortgat, Michael, 2010. "Categorial Type Logics." In Johan van Benthem and Alice ter Meulen (eds.), *Handbook of Logic and Language*, Amsterdam New York etc.: Elsevier Science B.V. Second edition.

Moot, R., 2002a. *Proof Nets for Linguistic Analysis*. Ph.D. thesis, UiL OTS, University of Utrecht.

Moot, Richard, 2002b. *Proof Nets for Linguistic Analysis*. Ph.D. thesis, University of Utrecht.

Morrill, Glyn, 2000. "Incremental Processing and Acceptability." *Computational Linguistics* 26:319–338.

Morrill, Glyn, 2007. "A Chronicle of Type Logical Grammar: 1935–1994." *Research on Language and Computation* 5(3):359–386.

Morrill, Glyn, 2011. *Categorial Grammar: Logical Syntax, Semantics, and Processing*. Oxford: Oxford University Press.

Morrill, Glyn V., 1994. *Type Logical Grammar: Categorial Logic of Signs*. Dordrecht, Boston, London: Kluwer Academic Publishers.

Muskens, Reinhard, 2001. "Categorial Grammar and Lexical-Functional Grammar." In Miriam Butt and Tracy Holloway King (eds.), *Proceedings of the LFG01 Conference*. CSLI Publications, 259–279.

Muskens, Reinhard, 2007. "Separating Syntax and Combinatorics in Categorial Grammar." *Research on Language and Computation* 5:267–285.

Oehrle, Richard T., 1999. "LFG as Labeled Deduction." In M. Dalrymple (ed.), *Semantics and Syntax in Lexical Functional Grammar*, Cambridge, MA: MIT Press. 319–357.

Oehrle, Richard T., 2011. "Multi-Modal Type-Logical Grammar." In Robert Borsley and Kersti Börjars (eds.), *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, New York: Blackwell.

Ouhalla, Jamal, 2001. "Parasitic Gaps and Resumptive Pronouns." In Peter Cullicover and Paul Postal (eds.), *Parasitic Gaps*, Cambridge: MIT Press. 147–180.

Pollard, Carl, 2004. "Type-Logical HPSG." In G. Jaeger, P. Monachesi, G. Penn, and S. Wintner (eds.), *Proceedings of Formal Grammar 2004 (Nancy)*. 107–124.

Pollard, Carl, 2008. "Hyperintensional Questions." In W. Hodges and R. de Queiroz (eds.), *Proceedings of the 15th Workshop on Logic, Language, Information, and Computation (WoLLIC '08)*. volume 5110 of *Springer Lecture Notes in Artificial Intelligence*, 261–274.

Pollard, Carl and Sag, Ivan, 1994. *Head Driven Phrase Structure Grammar*. CSLI/Chicago University Press, Chicago.

Ravi, Sujith, Baldridge, Jason, and Knight, Kevin, 2010. "Minimized Models and Grammar-Informed Initialization for Supertagging with Highly Ambiguous Lexicons." In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, 495–503.

Retoré, Christian and Stabler, Edward, 2004. "Generative Grammars in Resource Logics." *Research on Language and Computation* 2(1):3–25.

Ross, John Robert, 1967. *Constraints on Variables in Syntax*. Ph.D. thesis, MIT. Published as "Infinite Syntax!", Ablex, Norton, NJ. 1986.

Sag, Ivan A., Wasow, Tom A., and Bender, Emily, 2003. *Syntactic Theory: A Formal Introduction*. Stanford, California: Center for the Study of Language and Information, second edition.

Schönfinkel, Moses, 1924. "Über die Bausteine der mathematischen Logik." *Mathematische Annalen* 92(305-316).

Sgall, Petr, Hajičová, Eva, and Panevová, Jarmila, 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Dordrecht, Boston, London: D. Reidel Publishing Company.

Shieber, Stuart, 1985. "Evidence against the context-freeness of natural language." *Linguistics and Philosophy* 8:333–343.

Stabler, Edward, 1997. "Derivational minimalism." In Christian Retorè (ed.), *Logical Aspects of Computational Linguistics*, Springer. 68–95.

Steedman, Mark, 1985. "Dependency and Coordination in the Grammar of Dutch and English." *Language* 61:523–568.

Steedman, Mark, 1996a. *Surface Structure and Interpretation*. Cambridge Mass.: MIT Press. Linguistic Inquiry Monograph, 30.

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

Steedman, Mark, 2000a. "Implications of Binding for Lexicalized Grammars." In Anne Abeillé and Owen Rambow (eds.), *Tree Adjoining Grammars: Formalisms, Linguistic Analysis, and Processing*, Stanford: CSLI. 283–301.

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

Steedman, Mark, 2012. *Taking Scope*. MIT Press/Bradford Books.

Steedman, Mark and Baldridge, Jason, 2011. "Combinatory Categorial Grammar." In Robert Borsley and Kersti Börjars (eds.), *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, New York: Blackwell.

Szabolcsi, Anna, 1987. "Bound Variables in Syntax: Are There Any?" In Stokhof Gronendijk and Veltman (eds.), *Proceedings of the Sixth Amsterdam Colloquium*. Institute for Language, Logic, and Information, Amsterdam, 331–351.

Szabolcsi, Anna, 1992. "On Combinatory Grammar and Projection from the Lexicon." In Ivan Sag and Anna Szabolcsi (eds.), *Lexical Matters*, Stanford, CA: CSLI Publications. 241–268.

Szabolcsi, Anna, 2003. "Binding On the Fly: Cross-Sentential Anaphora in Variable-Free Semantics." In Richard Oehrle and Geert-Jan Kruijff (eds.), *Resource Sensitivity, Binding, and Anaphora*, Kluwer. 215–229.

Thomforde, Emily and Steedman, Mark, 2011. "Semi-supervised CCG Lexicon Extension." In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, 1246–1256.

Vadas, David and Curran, James R., 2008. "Parsing Noun Phrase Structure with CCG." In *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, 335–343.

van Benthem, Johann, 1989. "Logical Constants Across Varying Types." *Notre Dame Journal of Formal Logic* 30(3):315–342.

Vermaat, W., 1999. "Controlling Movement: Minimalism in a deductive perspective."

Vermaat, Willemijn, 2004. "The Minimalist Move Operation in a Deductive Perspective." *Research on Language and Computation* 2(1):69–85.

Vermaat, Willemijn, 2005. *The Logic of Variation: A cross-linguistic account of wh-question formation*. Ph.D. thesis, Utrecht University.

Vijay-Shanker, K. and Weir, David, 1990. "Polynomial Time Parsing of Combinatory Categorial Grammars." In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics, Pittsburgh*. 1–8.

Vijay-Shanker, K. and Weir, David, 1994. "The Equivalence of Four Extensions of Context-free Grammar." *Mathematical Systems Theory* 27:511–546.

Villavicencio, Aline, 2002. *The Acquisition of a Unification-Based Generalised Categorial Grammar*. Ph.D. thesis, University of Cambridge.

White, Michael, 2006. "Efficient Realization of Coordinate Structures in Combinatory Categorial Grammar." *Research on Language and Computation* 4(1):39–75.

Wood, Mary McGee, 1993. *Categorial Grammar*. Routledge.

Zeevat, Henk, Klein, Ewan, and Calder, Jo, 1987. "An Introduction to Unification Categorial Grammar." In N. Haddock et al. (ed.), *Edinburgh Working Papers in Cognitive Science, 1: Categorial Grammar, Unification Grammar, and Parsing*, University of Edinburgh. 195–222.

Zettlemoyer, Luke and Collins, Michael, 2007. "Online Learning of Relaxed CCG Grammars for Parsing to Logical Form." In *Proceedings of EMNLP-CoNLL 2007*.

Zhang, Yue, Blackwood, Graeme, and Clark, Stephen, 2012. "Syntax-Based Word Ordering Incorporating a Large-Scale Language Model." In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Avignon, France.