Normal-form parsing for Combinatory Categorial Grammars with generalized composition and type-raising

Julia Hockenmaier **Yonatan Bisk**
University of Illinois at Urbana-Champaign
{juliahmr, bisk1}@illinois.edu

Abstract

We propose and implement a modification of the Eisner (1996) normal form to account for generalized composition of bounded degree, and an extension to deal with grammatical type-raising.

1 Introduction

Combinatory Categorial Grammar (Steedman, 2000) is a linguistically expressive grammar formalism that has been used for many NLP applications, including wide-coverage parsing (Clark and Curran, 2007; Hockenmaier, 2003) and semantic interpretation (Curran et al., 2007), semantic role-labeling (Gildea and Hockenmaier, 2003; Boxwell et al., 2009), semantic parsing (Zettlemoyer and Collins, 2005) and natural language generation (Espinosa et al., 2008).

An essential feature of CCG is its flexible constituent structure, licensed by type-raising and composition rules which can create "non-standard" constituents such as *"John saw"*, or *"Mary talked to"*, required in constructions involving non-local dependencies, such as wh-extraction (Fig. 1) or right-node raising. Since *"John saw"* can now also be a constituent in *"John saw Mary"*, this leads to a combinatorial explosion of *spurious ambiguities*, i.e. multiple syntactic derivations of the same semantic interpretation (Wittenburg, 1986). This can create problems for applications based on CCG, e.g. for the induction of stochastic CCGs from text annotated with logical forms (Zettlemoyer and Collins, 2007), where spreading probability mass over equivalent derivations should be avoided. A number of *normal-form* (NF) parsing algorithms that aim to produce only one derivation per interpretation have been proposed (Wittenburg, 1986; Niv, 1994; Pareschi and Steedman, 1987; Hepple and Morrill, 1989; Eisner, 1996). Computationally, such algorithms are very attractive since they do not require costly semantic equivalence checks (Karttunen, 1989; Komagata, 2004) during parsing. Eisner's (1996) normal form is the most developed and well-known of these approaches, but is only defined for a variant of CCG where type-raising is a lexical operation and where the degree of composition is unbounded. Therefore, it and its equivalent reformulation by Hoyt and Baldridge (2008) in a multimodal variant of CCG are not safe (preserve all interpretations) and complete (remove all spurious ambiguities) for more commonly used variants of CCG. In particular, this NF is not safe when the degree of composition is bounded,[1] and not complete when type-raising is a grammatical operation. This paper defines a NF for CCG with bounded composition and grammatical type-raising.

2 Combinatory Categorial Grammar

In CCG, every constituent (*"John saw"*) has a syntactic category ($\mathsf{S/NP}$) and a semantic interpretation ($\lambda x.saw(john', x)$).[2] Constituents combine according to a small set of language-

[1] Although Eisner (1996, section 5) also provides a safe and complete parsing algorithm which can return non-NF derivations when necessary to preseve an interpretation if composition is bounded or the grammar is restricted in other (arbitrary) ways.

[2] More complex representations than simple predicate-argument structures are equally possible.

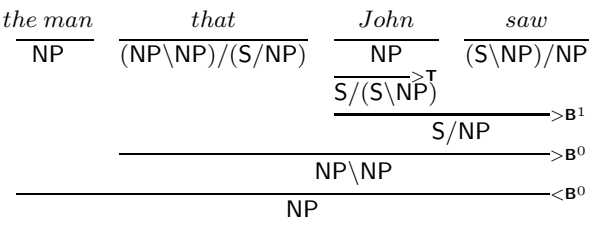


Figure 1: CCG derivations for wh-extraction

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 465–473,
Beijing, August 2010

Application	$(>)$	$X/Y: \lambda x.f(x)$	$Y: a$	$\Rightarrow X: f(a)$
	$(<)$	$Y: a$	$X\backslash Y: \lambda x.f(x)$	$\Rightarrow X: f(a)$
Composition	$(>\mathbf{B}^1)$	$X/Y: \lambda x.f(x)$	$Y/Z: \lambda y.g(y)$	$\Rightarrow X/Y: \lambda z.f(g(z))$
	$(<\mathbf{B}^1)$	$Y\backslash Z: \lambda y.g(y)$	$X\backslash Y: \lambda x.f(x)$	$\Rightarrow X\backslash Y: \lambda z.f(g(z))$
	$(>\mathbf{B}^1_\times)$	$X/Y: \lambda x.f(x)$	$Y\backslash Z: \lambda y.g(y)$	$\Rightarrow X\backslash Y: \lambda z.f(g(z))$
	$(<\mathbf{B}^1_\times)$	$Y/Z: \lambda y.g(y)$	$X\backslash Y: \lambda x.f(x)$	$\Rightarrow X/Y: \lambda z.f(g(z))$
	$(>\mathbf{B}^n)$	$X/Y: \lambda x.f(x)$	$Y\|Z_1\|...\|Z_n: \lambda z_n..z_1.g(z_1...z_n)$	$\Rightarrow X\|Z_1\|...\|Z_n: \lambda z_n...z_1.f(g(z_1...z_n))$
	$(<\mathbf{B}^n)$	$Y\|Z_1\|...\|Z_n: \lambda z_n..z_1.g(z_1...z_n)$	$X\backslash Y: \lambda x.f(x)$	$\Rightarrow X\|Z_1\|...\|Z_n: \lambda z_n...z_1.f(g(z_1...z_n))$
Typeraising	$(>\mathbf{T})$	For $X \in \mathcal{C}_{arg}$: $X: a$		$\Rightarrow T/_i(T\backslash_i X): \lambda f.f(a)$
	$(<\mathbf{T})$	For $X \in \mathcal{C}_{arg}$: $X: a$		$\Rightarrow T\backslash_i(T/_i X): \lambda f.f(a)$

Figure 2: CCG's combinatory rules.

independent combinatory rules (Fig. 2). The lexicon pairs words with categories and interpretations and is language-specific.

Syntax We distinguish atomic (S, NP, PP, etc.) from complex categories ((S\NP)/NP, N/N, etc.). A complex category of the form X/Y (or X\Y) represents a function which returns a *result* of type X when applied to an *argument* of type Y, which, in the case of a forward slash (/) has to follow the functor, and in the case of a backslash (\) has to preceed it. X and Y can themselves be complex again. We will use categories with vertical slashes when the direction of the slash does not matter, and may omit unnecessary parentheses (so X|Y|Z will represent (X\Y)/Z, (X\Y)\Z, ...). We will also use the shorthand $X|Y_{1..n}$ (or $X|\alpha$) to refer to a category with (possibly complex) result X and arguments $Y_1...Y_n$ (or an unspecified, possibly empty, list of arguments $\alpha = Y_{0...n}$, where $|\alpha| = n$) that can each appear with either type of slash.

Semantics If the category of a constituent is atomic (NP; S), its interpretation will also be atomic (*kim'*; *sleeps'(kim')*), and if the category is a functor of arity n ($X|Y_{1..n}$), the interpretation is a λ-expression $\lambda y_n..\lambda y_1\phi(y_1...y_n)$ of arity n.

The lexicon Each language defines a finite set of lexical category types $\mathcal{C}_{lex}$ (e.g. $(S\backslash NP)/NP$ is in the English lexicon, but $(S\backslash NP)\backslash NP$ is not) with maximal arity N_L. This defines a set of lexical argument category types $\mathcal{C}_{arg}$, consisting of all categories Y that are the argument of some lexical category $(X|Y)|\beta \in \mathcal{C}_{lex}$ (with $|\beta| \geq 0$). Since $\mathcal{C}_{lex}$ is finite, $\mathcal{C}_{arg}$ is strictly smaller than $\mathcal{C}_{lex}$ (and usually consists of basic categories such as NP, S, S\NP).

Combinatory Rules In addition to function application $(>, <)$, CCG has three kinds of combinatory rules (Fig. 2): harmonic function composition ($>\mathbf{B}^{(1)}$, $<\mathbf{B}^{(1)}$), crossing function composition ($>\mathbf{B}_\times$,$<\mathbf{B}_\times$) and type-raising ($>\mathbf{T}$, $<\mathbf{T}$). All rules take one or two *input* categories and yield one *output* category, and consist of a syntactic and a corresponding semantic operation. Composition also has *generalized* variants $>\mathbf{B}^n$, $<\mathbf{B}^n$ up to a fixed degree $N_\mathbf{B}$.[3] Composition of unbounded degree increases the generative capacity of CCG (Weir, 1988), and should be disallowed. Application $(>, <)$ can be seen as a special case of composition ($>\mathbf{B}^0$,$<\mathbf{B}^0$). When composing X|Y with Y|Z to X|Z, we call X|Y the *primary* input and Y|Z the *secondary* input. Harmonic composition allows associativity: the string A/B B/C C now has an alternative derivation where A/B and B/C compose into A/C, whereas crossing composition enables novel permutations, such as C A/B B\C.

Type-raising swaps the functor-argument relation. Although it is often assumed to take place in the lexicon, we will distinguish lexical categories (e.g. for quantifiers) that have the syntactic type of type-raised categories, but semantics that could not be obtained by type-raising a simple category from grammatically type-raised categories. We follow the common definition of CCG (Steedman, 2000) and allow only categories $X \in \mathcal{C}_{arg}$ to be type-raised.[4] Instantia-

[3] In $X|Y_{1..n}$ or $X|\alpha$=$X|Y_{1...|\alpha|}$, we do not assume the slash variable $| \in \{/, \backslash\}$ to be instantiated the same way for all Y_i. We will therefore only distinguish between forward and backward generalized composition $\mathbf{B}^{n>1}$.

[4] We stipulate that it may be further necessary to only allow those argument categories to type-raise that are not used to project unbounded dependencies, such as S/NP in

tions of the variable T should also be restricted to categories of finite arity $N_{\mathbf{T}}$ in oder to prevent an increase in generative capacity (Hoffman, 1995; Komagata, 1997). We refer to the arity of T as the degree of any particular instantation of $\mathbf{T}$. We follow Steedman (2000) and assume $N_{\mathbf{T}} = N_{\mathbf{B}}$.

Coordination requires a ternary rule (Φ) which can be binarized ($\Phi>$, $\Phi<$) to simplify parsing:[5]

$$\begin{array}{llll} (\Phi) & \mathsf{X} & \mathsf{conj}\ \mathsf{X} & \Rightarrow \mathsf{X} \\ (\Phi>) & \mathsf{X} & \mathsf{X[conj]} & \Rightarrow \mathsf{X} \\ (\Phi<) & \mathsf{conj} & \mathsf{X} & \Rightarrow \mathsf{X[conj]} \end{array}$$

Uses of type-raising and composition In English, type-raising and composition are required for wh-extraction and right node raising of arguments as well as so-called argument cluster coordination. In other languages, they are needed for scrambling and cross-serial dependencies.

It is important to note that when type-raising is *required*, it always occurs in tandem with composition. Since type-raising an argument Y to $\mathsf{X/(X\backslash Y)}$ and applying it to the functor $\mathsf{X\backslash Y}$ is semantically equivalent to applying $\mathsf{X\backslash Y}$ directly to Y, type-raising is never required when function application can be used instead. That is, in all cases, a type-raised argument must be composed with another constituent, usually the original functor (head). Only in argument-cluster coordination will the type-raised element be composed with a non-head constituent. In the latter case, coordination will be required before the argument cluster can be combined with the head. Composition without type-raising may occur, e.g. for adjuncts, which have categories $\mathsf{X|X}$, but may modify a constituent with category $\mathsf{X}|\alpha$.

Restrictions on type-raising and composition In order to prevent overgenerations of the form *"John speaks because Chinese, he enjoys Beijing."*, we assume a variant of CCG in which forward crossing composition $>\mathbf{B}^1_\times$ (e.g. of *because*:$\mathsf{(S/S)/S}$) into the result of backward type-raising $<\mathbf{T}$ (e.g. *Chinese*:$\mathsf{S\backslash(S/NP)}$, and, similarly, $<\mathbf{B}^x$ into the result of $>\mathbf{T}$, are disallowed.

$\mathsf{(NP\backslash NP)/(S/NP)}$ for English object relative pronouns.

[5]Here, X needs to be restricted to a finite set of categories (Weir, 1988). In multimodal CCG, conjunction have categories of the form $\mathsf{(X_\star\backslash_\star X)/_\star X}$, i.e. must apply to their argument

Punctuation and Type-changing rules CCGbank (Hockenmaier and Steedman, 2007) uses special punctuation rules such as $\mathsf{S}\,.\Rightarrow\mathsf{S}$ or $,\,\mathsf{NP\backslash NP}\Rightarrow\mathsf{NP\backslash NP}$, and a small number of (non-recursive) type-changing rules (with idiosyncratic semantics) such as $\mathsf{N}\Rightarrow\mathsf{NP}$ (for determiner-less NPs) or $\mathsf{S[pss]\backslash NP}\Rightarrow\mathsf{NP\backslash NP}$ (for complex adjuncts, here passive VPs being used as NP postmodifiers):

$$\begin{array}{lll} \text{Punctuation} & (>\mathbf{P}) & \mathsf{X}{:}\phi\ [.,;]\Rightarrow\mathsf{X}{:}\phi \\ & (<\mathbf{P}) & [.,;]\ \mathsf{X}{:}\phi\Rightarrow\mathsf{X}{:}\phi \\ \text{TypeChanging} & (\mathbf{TCR}) & \mathsf{X}{:}\phi\Rightarrow\mathsf{Y}{:}\psi(\phi) \end{array}$$

CCG parsing CCG can be parsed with a bottom-up CKY-like algorithm (Shieber et al., 1995; Steedman, 2000), which differs from standard CKY in that it requires one (or two) unary completion steps in each cell to deal with type-raising (and type changing).[6] Chart items are of the form $\langle \mathsf{X}, i, j\rangle$, where X is a category, and the indices i and j represent the span of the item. Interpretations need only to be constructed for complete derivations when unpacking the chart.

3 The Eisner normal form

The Eisner normal form Eisner (1996) presents a normal-form parsing algorithm for CCG without grammatical type raising (where the lexicon may still contain categories like $\mathsf{S/(S\backslash NP)}$, but there is no combinatory rule that changes a complex (derived) NP to e.g. $\mathsf{S/(S\backslash NP)}$). He proves that his algorithm finds only one canonical derivation for each semantic interpretation of an input string consisting of a sequence of words and their lexical categories. Since the presence of both pre- and postmodifiers (as in *"intentionally knock twice"*[7]) introduces a genuine ambiguity, Eisner proves that the only kind of spurious ambiguity that can arise in his variant of CCG is due to associative chains of composition such as $\mathsf{A/B\ B/C\ C/D}$ or $\mathsf{A/B\ B/C\ C\backslash D}$, which can be derived as either

[6]Since composition allows the arity of derived ($\approx$ non-terminal) CCG categories to grow with the length of the input string, worst-case complexity of this naive algorithm is exponential. (Vijay-Shanker and Weir, 1993)'s $O(n^6)$ algorithm has a more compact representation of categories.

[7]This can mean $\lambda x.intentionally'(twice'(knock'(x)))$ or $\lambda x.twice'(intentionally'(knock'(x)))$.

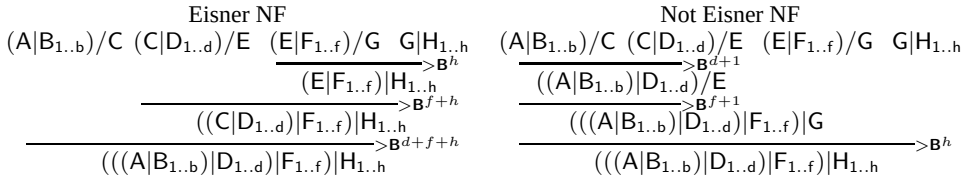


Figure 3: Eisner NF and generalized composition $\mathbf{B}^{n>1}$

Left branching	Right branching		
$>\mathbf{B}^0(>\mathbf{B}^{m+1},...)$	$\Rightarrow >\mathbf{B}^{m\geq 0}(...,>\mathbf{B}^0)$	A/B (B|$D_{0..m}$)/C C	$m \geq 0$
$>\mathbf{B}^1(>\mathbf{B}^{m\geq 1},...)$	$\Rightarrow >\mathbf{B}^{m\geq 1}(...,>\mathbf{B}^1)$	A/B (B|$C_{1...m-1}$)/D D/E	$m \geq 1$
$>\mathbf{B}^{n\geq 1}(>\mathbf{B}^1,...)$	$\Rightarrow >\mathbf{B}^n(...,>\mathbf{B}^{m=n})$	A/B B/C C/$D_{1..n}$	$m = n \geq 1$
$\emptyset$	$\not\Leftarrow >\mathbf{B}^{n>1}(...,>\mathbf{B}^{m>n})$	A/(B|$D_{1..k}$) B/C ((C|$D_{1..k}$)|$E_{1..n}$	$m > n > 1$
$>\mathbf{B}^m(>\mathbf{B}^k,...)$	$\Leftarrow >\mathbf{B}^{n>1}(...,>\mathbf{B}^{1<m<n})$	A/B (B|$C_{1..k-1}$)/D D|$E_{1..m}$	$n > m > 1$

Figure 4: Associative composition chains: our NF disallows the grayed-out derivations.

$>\mathbf{B}$ $(...,>\mathbf{B}$ $)$ or $>\mathbf{B}$ $(>\mathbf{B}$ $,\_)$. This is eliminated by the following constraint:

Eisner NF Constraint 1. *The output* $X|\alpha$ *of forward composition* $>\mathbf{B}^{n>0}$ *cannot be the primary input to forward application or composition* $>\mathbf{B}^{m\geq 0}$. *The output of* $<\mathbf{B}^{n>0}$ *cannot be the primary input to* $<\mathbf{B}^{m\geq 0}$.

This can be implemented by a ternary feature $H_E \in \{>\mathbf{B}^n,\ <\mathbf{B}^n,\ \emptyset\}$ and chart items of the form $\langle X,\ H_E,\ i,\ j\rangle$ where $H_E = >\mathbf{B}^n$ (or $<\mathbf{B}^n$) if X was produced by the corresponding composition rule (for any $n > 0$) and $\emptyset$ otherwise.

4 A new normal form for CCG

4.1 Generalized composition

Eisner NF and generalized composition Unboundedly long sequences of generalized composition are required e.g. for Dutch verb clusters that give rise to cross-serial dependencies ($N_1...N_nV_1...V_n$ with N_i the argument of V_i). These can be obtained through standard bounded-degree compositions, but the Eisner NF produces a derivation that requires compositions of unbounded degree (Fig. 3). Although this is allowed in the variant of CCG Eisner considers, compositions of unbounded degree are usually disallowed because they increase the generative capacity of CCG (Weir, 1988). We stipulate that the NF of any derivation τ should not require composition rules of higher degree than τ itself. Note that the output of function application ($\mathbf{B}^0$) always has lower arity than its functor; the output of regular composition ($\mathbf{B}^1$) has the same arity as its primary functor, but the output of generalized composition ($\mathbf{B}^{n>1}$) has an arity that is $n-1$ higher than that of the primary functor. $\mathbf{B}^{n>1}$ therefore requires a different treatment.

Our reformulation of the Eisner NF Associative composition chains for constituents A B C can lead to spurious ambiguity if both a left-branching $>\mathbf{B}^n(>\mathbf{B}^m$(A B) C) and a right-branching $>\mathbf{B}^{n'}$(A $>\mathbf{B}^{m'}$(B C)) are possible and lead to the same interpretation. Figure 4 illustrates all possible cases consisting of three constituents. In most cases, the right-branching (Eisner NF) derivation is to be preferred. For generalized composition $>\mathbf{B}^{n>1}$, $>\mathbf{B}^{m>1}$, left-branching $>\mathbf{B}^{n>1}(>\mathbf{B}^{m>1},...)$ is always allowed, but right-branching $>\mathbf{B}^n(...,>\mathbf{B}^m)$ is only allowed if $m \geq n$.

NF Constraint 1 ($\mathbf{B}^0$ and $\mathbf{B}^{n\geq 1}$)**.** *The output of* $>\mathbf{B}^{n\geq 1}$ *(resp.* $<\mathbf{B}^{n\geq 1}$*) cannot be primary functor for* $>\mathbf{B}^{n\leq 1}$ *(resp.* $<\mathbf{B}^{n\leq 1}$*).*

NF Constraint 2 ($\mathbf{B}^1$ and $\mathbf{B}^{n\geq 1}$)**.** *The output of* $>\mathbf{B}^1$ *(resp.* $<\mathbf{B}^1$*) cannot be primary functor for* $>\mathbf{B}^{n\geq 1}$ *(resp.* $<\mathbf{B}^{n\geq 1}$*).*

NF Constraint 3 ($\mathbf{B}^{n>1}$ and $\mathbf{B}^{m>1}$)**.** *The output of* $>\mathbf{B}^m$ *(resp.* $<\mathbf{B}^m$*) cannot be secondary functor for* $>\mathbf{B}^{n>m}$ *(resp.* $<\mathbf{B}^{n>m}$*).*

4.2 Grammatical type-raising

Eisner NF and type-raising Figure 5 illustrates a spurious ambiguity arising through type-

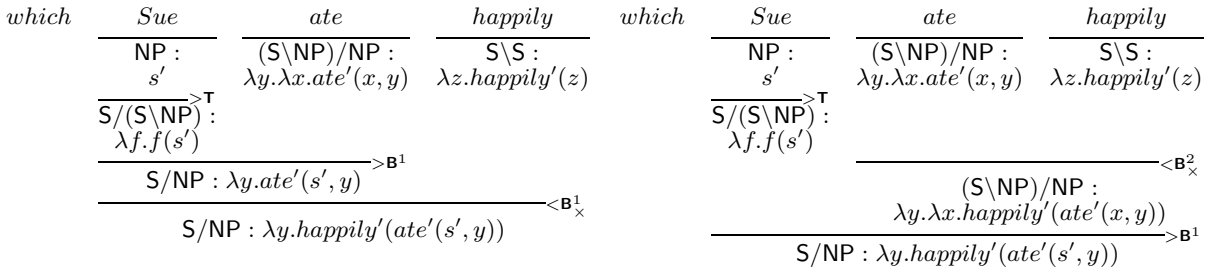
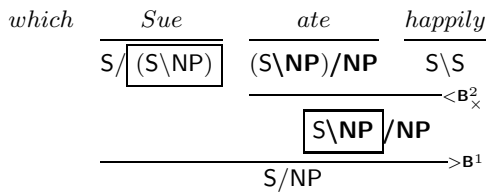


Figure 5: The Eisner NF allows spurious ambiguity arising due to type-raising

raising that the Eisner NF does not exclude.[8] Here two derivations can be obtained because the result of combining the adverb with the subject-verb cluster is no longer the output of a forward composition, and can therefore apply to the object. The derivations are semantically equivalent: although type-raising reverses the syntactic functor-argument relation, a type-raised argument applied to a predicate returns the same interpretation as when the predicate is applied directly to the original. But Eisner treats $\mathsf{S/(S\backslash NP)}$ as a category with semantics $\lambda x.\phi(x)$, in which case the derivations yield indeed different scope relations. Eisner's analyis is correct for certain classes of words which have lexical categories that appear like type-raised categories, but have a different interpretation from that of categories obtained by type-raising. These are usually scope-bearing elements, such as the universal quantifer *every* ($\mathsf{(S/(S\backslash NP))/N} : \lambda\mathsf{P}\lambda\mathsf{Q}\forall\mathsf{x}\mathsf{P}(\mathsf{x}) \rightarrow \mathsf{Q}(\mathsf{x})$), and there may not be a single derivation which captures all semantic interpretations. Lexicalized pseudo-type-raising therefore needs to be distinguished from grammatical type-raising.

Our extension of the (modified) Eisner NF In Fig. 5, Eisner NF licenses two derivations. Both contain an instance of composition in which the type-raised argument is the primary component. In the analysis in which this is the second derivation step, the canceled part of this $<\mathbf{B}^2$ composition (boxed) contains a category ($\backslash\mathsf{NP}$) that was part of the argument output of the first $>\mathbf{B}^1$ composition (bold-faced):



Our NF will eliminate derivations of this type and prefer the other, lower-degree derivation. We stipulate that the spurious ambiguities that arise through type-raising and composition can be eliminated through the following rule:

NF Constraint 4 (**T** and $\mathbf{B}^{n>0}$)**.** *The output of* $>$**T** *cannot be primary input to* $>\mathbf{B}^{n>0}$ *if the secondary input is the output of* $<\mathbf{B}^{m>n}$*. The output of* $<$**T** *cannot be primary input in* $<\mathbf{B}^{n>0}$ *if the secondary input is the output of* $>\mathbf{B}^{m>n}$*.*

We also stipulate that a type-raised $\mathsf{T/(T\backslash X)}$ cannot be used as a functor in application (since $\mathsf{T\backslash X}$ could always apply directly to X).

NF Constraint 5 (**T** and $\mathbf{B}^0$)**.** *The output of forward (or backward) type-raising* $>$**T** *(resp.* $<$**T** *) cannot be the functor in application* $>$ *(resp.* $<$*).*

Additional spurious ambiguities arise through the interaction of type-raising and coordination: Since any category can be coordinated, we can either coordinate X and then type-raise the coordinated X to $\mathsf{T/(T\backslash X)}$, or we can first type-raise each conjunct to $\mathsf{T/(T\backslash X)}$ and then conjoin. Since nonsymmetric coordinations of an argument-adjunct cluster and a single argument (as in *eats ((pizza for lunch) and pasta)*) require type-raising before coordination, we formulate the following rule to eliminate interactions between type-raising and coordination:

NF Constraint 6 (**T** and Φ)**.** *The result of coordination* Φ *cannot be type-raised.*

[8] We have chosen a slighly unusual adverb category to illustrate a general problem.

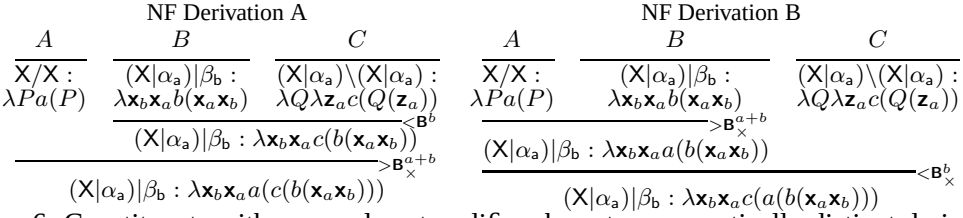


Figure 6: Constituents with pre- and postmodifiers have two semantically distinct derivations

Punctuation and Type-changing rules Punctuation results in spurious ambiguities, either when a constituent X has both an initial and a final punctuation mark (e.g. a comma), or when it has an initial (final) punctuation mark and a final (initial) modifier. The first case is easy to fix by disallowing the output of , $\mathsf{X} \Rightarrow \mathsf{X}$ to be the input of X , $\Rightarrow \mathsf{X}$. The latter could be eliminated by disallowing the output X of right-recursive (left-recursive) punctuation rule to be secondary input to any left-recursive (right-recursive) application or composition rule (e.g. $\mathsf{X}\ \ \mathsf{X}\backslash\mathsf{X} \Rightarrow \mathsf{X}$).[9]

Implementation Our normal-form constraints can be implemented in a bottom-up parser with items of the form $\langle \mathsf{X}, \mathcal{C}, i, j \rangle$, with

$$\mathcal{C} \in \{>, >\mathbf{B}\ 1, >\mathbf{B}\ 2, ..., >\mathbf{B}^n; <, <\mathbf{B}\ 1, <\mathbf{B}\ 2, ..., <\mathbf{B}^n; \\ >\mathbf{T}\ , <\mathbf{T}\ , >\mathsf{Pct}, <\mathsf{Pct}, \Phi>, \Phi<, \mathsf{TCR}\}$$

4.3 Is our normal form safe and complete?

Here we sketch the beginnings of a proof that our algorithm allows one and only one syntactic derivation per semantic interpretation for the version of CCG we consider. We first examine all cases of two adjacent constituents A, B which must combine into a category C:

Functor X/Y **and argument** Y **combine to** X The functor must apply to the argument. The argument could type-raise, but then cannot apply.

Functor $\mathsf{X}/\mathsf{Y}|\alpha$ **and argument** Y **combine to** $\mathsf{X}|\alpha$ The functor cannot apply to the argument. The argument must type-raise to $\mathsf{X}\backslash(\mathsf{X}/\mathsf{Y})$, and can then backward-compose into the functor.

Functor X/X **and** $\mathsf{X}\backslash\mathsf{X}$ **can combine to** X/X **or** $\mathsf{X}\backslash\mathsf{X}$ This is not a spurious ambiguity, since the output categories are different.

[9] If punctuation can be used both with X and Y, it also interacts with type-changing rules $\mathsf{X} \Rightarrow \mathsf{Y}$. Our current implementation does not deal with this case.

Functor $\mathsf{X}|\mathsf{Y}$ **and** $\mathsf{Y}|\mathsf{Z}$ **combine to** $\mathsf{X}|\mathsf{Z}$ Our reformulation of Eisner's NF eliminates spurious ambiguities that are due to such associative composition chains. This covers not only argument clusters (which must compose), but also ambiguous cases where one constituent (e.g. Y/Z with $\alpha = \epsilon$) is the argument of the first (X/Y), and either takes the third (Z) as its own argument or is modified by the third $\mathsf{Y}\backslash\mathsf{Y}$ (there are, of course, other arrangements of such categories which are not ambiguous, e.g. $\mathsf{X}/\mathsf{Y}\ \mathsf{Z}\ \mathsf{Y}\backslash\mathsf{Z}$.

We now focus our attention on the ternary cases in which one of the constituents is a head (predicate), and the other two are either its arguments or modifiers. The counterexample to Eisner's normal-form algorithm shows that there is at least one additional kind of spurious ambiguity that arises when there are three adjacent constituents A, B, C and both A and C can compose into B. There are three cases: 1) A and C are both modifiers of B, 2) one of A or C is a modifier of B, the other is an argument of B, and 3) A and C are both arguments of B. Only 1) is a real ambiguity, but the other cases are instances of spurious ambiguity which our NF eliminates.

Argument Y**, head** $(\mathsf{X}\backslash\mathsf{Y})/\mathsf{Z}$ **and argument** Z **combine to** X In the NF derivation, the head applies first to the Z, than to Y. All other derivations are blocked, either because type-raised categories cannot apply, or because the output of composition cannot apply.

Modifier X/X**, head** $(\mathsf{X}|\alpha)|\beta$ **and modifier** $(\mathsf{X}|\alpha)\backslash(\mathsf{X}|\alpha)$ **combine to** $(\mathsf{X}|\alpha)|\beta$ (Fig. 4.2). This is the *"intentionally knock twice"* example. The derivations have different semantics.

Argument Y**, head** $((\mathsf{X}|\alpha)\backslash\mathsf{Y})|\beta$**, and modifier** $\mathsf{X}\backslash\mathsf{X}$ **combine to** $(\mathsf{X}|\alpha)|\beta$ (Fig. 7). If there is an ambiguity, B must have a category of the form

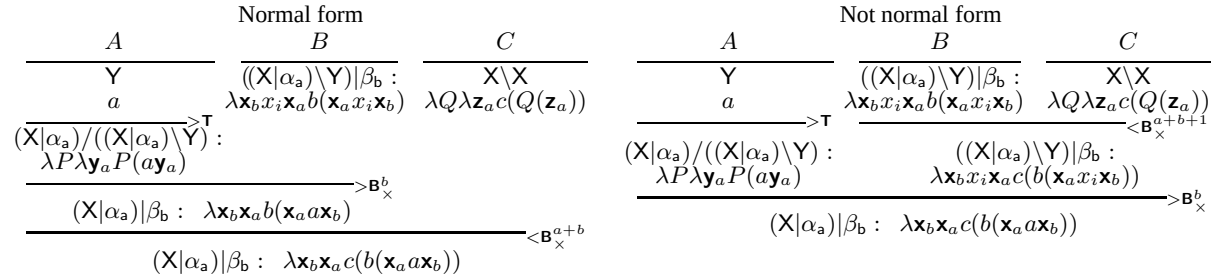


Figure 7: Argument Y, head $((\mathsf{X}|\alpha_\mathsf{a})\backslash\mathsf{Y})|\beta_\mathsf{b}$, and modifier $\mathsf{X}\backslash\mathsf{X}$ combine to $(\mathsf{X}|\alpha_\mathsf{a})|\beta_\mathsf{b}$

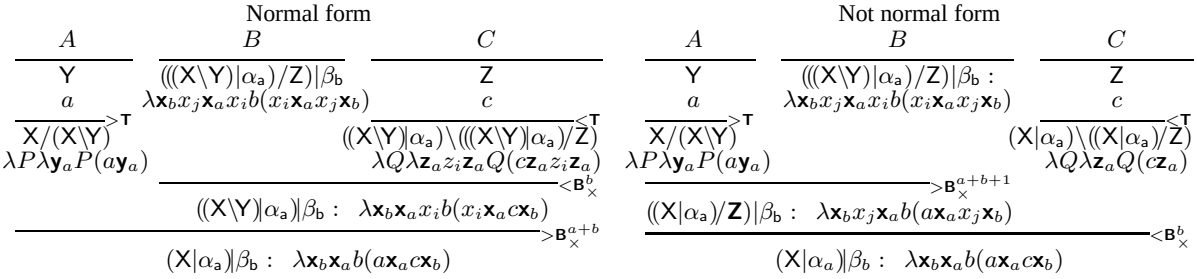


Figure 8: Argument Y, head $(((\mathsf{X}\backslash\mathsf{Y})|\alpha)/\mathsf{Z})|\beta$ and argument Z combine to $(\mathsf{X}|\alpha)|\beta$

$((\mathsf{X}|\alpha)\backslash\mathsf{Y_i})|\beta$ (with X possibly complex and α, β possibly empty), and C must have a category of the form $\mathsf{X}\backslash\mathsf{X}$. We obtain the NF derivation by first combining head and argument, followed by the modifier. The other derivation violates the NF constraints.

Argument Y**, head** $(((\mathsf{X}\backslash\mathsf{Y})|\alpha)/\mathsf{Z})|\beta$ **and argument** Z **combine to** $(\mathsf{X}|\alpha)|\beta$ (Fig. 8) The derivation in which Z composes first is in NF. The derivation in which the Y combines first with the head is blocked.

Arguments $\mathsf{Y_A}$**,** $\mathsf{Y_B}$**, head** $(((\mathsf{X}\backslash\mathsf{Y_1})|\alpha)\backslash\mathsf{Y_2})|\beta$ **combine to** $(\mathsf{X}|\alpha)|\beta$ There are two readings: standard ($\mathsf{Y_A}$:=$\mathsf{Y_1}$, $\mathsf{Y_B}$:=$\mathsf{Y_2}$), and scrambled ($\mathsf{Y_A}$:=$\mathsf{Y_2}$, $\mathsf{Y_B}$:=$\mathsf{Y_1}$). If α and β are empty, function application is sufficient for the standard reading, and our NF constraint 1 excludes the 'argument cluster' derivation in which both $\mathsf{Y_A}$ and $\mathsf{Y_B}$ type-raise, compose and then apply to the head. Otherwise, at least one of the arguments has to type-raise and compose into the head. If both α and β are non-empty, each interpretation has only one derivation in which the type-raised $\mathsf{Y_A}$ composes into the output of the composition of the type-raised $\mathsf{Y_B}$ with the head. Since the degree of the second composition is lower than the first, this is allowed by our NF constraint 2.

Argument $\mathsf{Y_A}$ **and heads** $(((\mathsf{X}\backslash\mathsf{Y_1})|\alpha)/\mathsf{Z}$ **and** $((\mathsf{Z}|\beta)\backslash\mathsf{Y_2})|\gamma$ **combine to** $(((\mathsf{X}|\alpha)|\beta)\backslash\mathsf{Y_2})|\gamma$ **or to** $(((\mathsf{X}|\backslash\mathsf{Y_1}\alpha)|\beta)|\gamma$ There are two readings: standard ($\mathsf{Y_A}$:=$\mathsf{Y_1}$) or scrambled ($\mathsf{Y_A}$:=$\mathsf{Y_2}$). Depending on the maximal degree n of $\mathbf{B}^n$ allowed by the grammar, the standard reading one can either be obtained by type-raising $\mathsf{Y_A}$ and composing into the first head (allowed by our NF) or by first composing the two heads and then composing the type-raised $\mathsf{Y_A}$ into the cluster (allowed by Eisner, but not by us). The second reading requires the heads to compose and then $\mathsf{Y_A}$ to apply or compose (depending on the arity of γ), which is allowed by our NF constraint 2 because the degree of this second composition is lower than that of the first.

Our NF and the bound $N_\mathbf{T}$ **on type-raising** If $\mathsf{X}\backslash\mathsf{X}$ in Fig. 7 is replaced with a (non-type-raised) category $\mathsf{Z}\backslash\mathsf{X}$ (for $\mathsf{Z} \neq \mathsf{X}$), the non-NF derivation requires $\mathbf{T}^{|Z|+a}$, whereas the NF-derivation requires $\mathbf{T}^{|X|+a}$. If we stipulate a finite bound $N_\mathbf{T}$ on the degree of type-raising, and if $|X| > |Z|$ and $|X| + a > N_\mathbf{T}$, our NF cannot be derived anymore. If such $\mathsf{Z}\backslash\mathsf{X}$ (with $\mathsf{X} \in \mathcal{C}_{arg}$) can be derived from the lexicon, our NF requires therefore a potentially unbounded degree of type-raising. The **T**-degree

	Sentence length l=15...30			
	15	20	25	30
No NF (total #derivs)	4.13E6	5.66E8	3.06E11	1.59E14
Eisner **B**	18.92%	9.05%	3.63%	2.14%
Our **B**	18.38%	8.97%	3.60%	2.02%
Our **B** , **T**	2.92%	1.22%	0.37%	0.10%
Our full NF	2.60%	0.93%	0.33%	0.09%

(a) Median % of allowed derivations

	Sentence length l= 30			
	Min	Mean	Median	Max
No NF	5.99E9	8.19E15	1.59E14	2.61E17
Eisner **B**	1.60%	2.68%	2.14%	2.76%
Our **B**	1.57%	2.49%	2.02%	2.69%
Our **B** ,**T**	0.64%	0.07%	0.10%	0.05%
Our full NF	0.53%	0.06%	0.09%	0.05%

(b) Statistics on the % of allowed derivations

Figure 9: Experimental results: the effect of different normal forms on the number of derivations

of the non-NF derivation in Fig. 8 is also one less than that of the NF derivation, but its **B**-degree is increased by one, so for $N_{\mathbf{T}} = N_{\mathbf{B}}$ either both derivations are possible or neither.

What remains to be proven is that we have considered all cases of spurious ambiguity involving three constituents, and that all cases of spurious ambiguity that arise for more than three constituents reduce to these cases.

5 The effects of normal form parsing

We now illustrate the impact of the different normal form variants on a small, restricted, grammar. We define a set of atomic categories, a set of lexical categories (up to a fixed arity N_{Lex}), and compile out all possible rule instantiations (including compositions up to a fixed degree $N_{|B}$) that generate categories up to a fixed arity N_{cat}[10]

The effect of different normal forms This experiment is intended to examine how normal form parsing might reduce spurious ambiguity for actual grammars, e.g. for unsupervised estimation of stochastic CCGs. We created a small English grammar with atomic categories $\mathsf{S}, \mathsf{NP}, \mathsf{N}, \mathsf{conj}, ., , ;$ and 47 lexical categories using $N_{Lex} = 3$, $N_{\mathbf{B}} = 3$, $N_{Cat} = 15$. There are two type-changing rules ($\mathsf{N} \Rightarrow \mathsf{NP}$ and $\mathsf{S/NP} \Rightarrow \mathsf{NP\backslash NP}$). We accept derivations of S, NP and $\mathsf{S\backslash NP}$. The $\mathsf{T|X}$ in **T** has to be a lexical category. Our lexical categories are divided into disjoint sets of adjuncts of the form $\mathsf{X|X}$ and $\mathsf{(X|X)|Y}$, head (both atomic and complex), and punctuation and conjunction categories. The comma can act as a conjunction or to set off modifiers (requiring punctuation rules of the form $\mathsf{X|X}$, $\Rightarrow$ $\mathsf{X|X}$ and , $\mathsf{X|X} \Rightarrow \mathsf{X|X}$). We furthermore define coarse-grained parts of speech (noun, verb, function word, conj, other) and decide for each part of speech which lexical categories it can take. We compare different NF settings for sentences of lengths 15–30 from Europarl (Koehn, 2005). At each length, we compare 100 sentences that our grammar can parse. All NFs can parse all sentences the full grammar can parse. Results (Fig. 9(a)) show that our NF reduces the number of derivations significantly over Eisner's NF, even though our (full) grammar only allows a restricted set of type-raising rules. Fig. 9(b) illustrates the combinatorial explosion of spurious derivations as the sentence length increases.

6 Conclusions

We have proposed a modification and extension of Eisner (1996)'s normal form that is more appropriate for commonly used variants of CCG with grammatical type-raising and generalized composition of bounded degree, as well as some non-combinatory extensions to CCG. Our experiments indicate that incorporating normal form constraints to deal with grammatical type-raising drastically reduces the number of derivations. We have sketched the outline of a proof that our normal form is safe and complete for the variant of CCG we consider, althoug we have seen that under certain circumstances, type-raising of unbounded degree may be required. Future work will investigate this issue further, and will also aim to turn our informal arguments about the adequacy of our approach into a full proof, and provide more experiments on a wider range of grammars and languages.

[10]The restriction of categories to a fixed arity means that we could generate cross-serial dependencies $N_1...N_nV_1...V_n$ only up to $n = A_{cat}$.

References

Boxwell, Stephen, Dennis Mehay, and Chris Brew. 2009. Brutus: A semantic role labeling system incorporating CCG, CFG, and dependency features. In *Proceedings of the 47th ACL/4th IJCNLP*, pages 37–45.

Clark, Stephen and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Curran, James, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and boxer. In *Proceedings of the 45th ACL Companion Volume (Demo and Poster Sessions)*, pages 33–36, Prague, Czech Republic.

Eisner, Jason. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th ACL*, pages 79–86, Santa Cruz, CA.

Espinosa, Dominic, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, Ohio.

Gildea, Daniel and Julia Hockenmaier. 2003. Identifying semantic roles using Combinatory Categorial Grammar. In *Proceedings of EMNLP*, Sapporo, Japan.

Hepple, Mark and Glyn Morrill. 1989. Parsing and derivational equivalence. In *Proceedings of the Fourth EACL*, pages 10–18, Manchester, UK.

Hockenmaier, Julia and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

Hockenmaier, Julia. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

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.

Hoyt, Frederick and Jason Baldridge. 2008. A logical basis for the D combinator and normal form in CCG. In *Proceedings of ACL-08: HLT*, pages 326–334, Columbus, Ohio.

Karttunen, Lauri. 1989. Radical lexicalism. In Baltin, M.R. and A.S. Kroch, editors, *Alternative Conceptions of Phrase Structure*. Chicago University Press, Chicago.

Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. In *10th MT Summit*, pages 79–86, Phuket, Thailand.

Komagata, Nobo. 1997. Generative power of CCGs with generalized type-raised categories. In *ACL35/EACL8 (Student Session)*, pages 513–515.

Komagata, Nobo. 2004. A solution to the spurious ambiguity problem for practical combinatory categorial grammar parsers. *Computer Speech & Language*, 18(1):91 – 103.

Niv, Michael. 1994. A psycholinguistically motivated parser for CCG. In *Proceedings of The 32nd ACL*, Las Cruces, NM, pages 125–132.

Pareschi, Remo and Mark Steedman. 1987. A lazy way to chart parse with categorial grammars. In *Proceedings of the 25th ACL*, pages 81–88, Stanford, CA.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36, July–August.

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

Vijay-Shanker, K and David J Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

Weir, David. 1988. *Characterising Mildly Context-sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania. Tech. Report CIS-88-74.

Wittenburg, Kent B. 1986. *Natural Language Parsing with Combinatory Categorial Grammar in a Graph-Unification Based Formalism*. Ph.D. thesis, University of Texas at Austin.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st UAI*, pages 658–666, Edinburgh, UK.

Zettlemoyer, Luke and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL*, pages 678–687, Prague, Czech Republic.

Acknowledgements

We would like to thank Mark Steedman for helpful discussions, and Jason Eisner for his very generous feedback which helped to greatly improve this paper. All remaining errors and omissions are our own responsibility. J.H is supported by NSF grant IIS 08- 03603 INT2-Medium.