The Importance of Rule Restrictions in CCG

Marco Kuhlmann
Dept. of Linguistics and Philology
Uppsala University
Uppsala, Sweden

Alexander Koller
Cluster of Excellence
Saarland University
Saarbrücken, Germany

Giorgio Satta
Dept. of Information Engineering
University of Padua
Padua, Italy

Abstract

Combinatory Categorial Grammar (CCG) is generally construed as a fully lexicalized formalism, where all grammars use one and the same universal set of rules, and cross-linguistic variation is isolated in the lexicon. In this paper, we show that the weak generative capacity of this 'pure' form of CCG is strictly smaller than that of CCG with grammar-specific rules, and of other mildly context-sensitive grammar formalisms, including Tree Adjoining Grammar (TAG). Our result also carries over to a multi-modal extension of CCG.

1 Introduction

Combinatory Categorial Grammar (CCG) (Steedman, 2001; Steedman and Baldridge, 2010) is an expressive grammar formalism with formal roots in combinatory logic (Curry et al., 1958) and links to the type-logical tradition of categorial grammar (Moortgat, 1997). It has been successfully used for a wide range of practical tasks, such as data-driven parsing (Hockenmaier and Steedman, 2002; Clark and Curran, 2007), wide-coverage semantic construction (Bos et al., 2004), and the modelling of syntactic priming (Reitter et al., 2006).

It is well-known that CCG can generate languages that are not context-free (which is necessary to capture natural languages), but can still be parsed in polynomial time. Specifically, Vijay-Shanker and Weir (1994) identified a version of CCG that is weakly equivalent to Tree Adjoining Grammar (TAG) (Joshi and Schabes, 1997) and other mildly context-sensitive grammar formalisms, and can generate non-context-free languages such as $a^n b^n c^n$. The generative capacity of CCG is commonly attributed to its flexible *composition* rules, which allow it to model more complex word orders that context-free grammar can.

The discussion of the (weak and strong) generative capacity of CCG and TAG has recently been revived (Hockenmaier and Young, 2008; Koller and Kuhlmann, 2009). In particular, Koller and Kuhlmann (2009) have shown that CCGs that are *pure* (i.e., they can only use generalized composition rules, and there is no way to restrict the instances of these rules that may be used) and *first-order* (i.e., all argument categories are atomic) can *not* generate $a^n b^n c^n$. This shows that the generative capacity of at least first-order CCG crucially relies on its ability to restrict rule instantiations, and is at odds with the general conception of CCG as a fully lexicalized formalism, in which all grammars use one and the same set of universal rules. A question then is whether the result carries over to pure CCG with higher-order categories.

In this paper, we answer this question to the positive: We show that the weak generative capacity of general pure CCG is still strictly smaller than that of the formalism considered by Vijay-Shanker and Weir (1994); composition rules can only achieve their full expressive potential if their use can be restricted. Our technical result is that every language L that can be generated by a pure CCG has a context-free sublanguage $L' \subseteq L$ such that every string in L is a permutation of a string in L', and vice versa. This means that $a^n b^n c^n$, for instance, cannot be generated by pure CCG, as it does not have any (non-trivial) permutation-equivalent sublanguages. Conversely, we show that there are still languages that can be generated by pure CCG but not by context-free grammar.

We then show that our permutation language lemma also holds for pure *multi-modal* CCG as defined by Baldridge and Kruijff (2003), in which the use of rules can be controlled through the lexicon entries by assigning types to slashes. Since this extension was intended to do away with the need for grammar-specific rule restrictions, it comes as quite a surprise that pure multi-modal

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 534–543,
Uppsala, Sweden, 11-16 July 2010.

CCG in the style of Baldridge and Kruijff (2003) is still less expressive than the CCG formalism used by Vijay-Shanker and Weir (1994). This means that word order in CCG cannot be fully lexicalized with the current formal tools; some ordering constraints must be specified via language-specific combination rules and not in lexicon entries. On the other hand, as pure multi-modal CCG has been successfully applied to model the syntax of a variety of natural languages, another way to read our results is as contributions to a discussion about the exact expressiveness needed to model natural language.

The remainder of this paper is structured as follows. In Section 2, we introduce the formalism of *pure* CCG that we consider in this paper, and illustrate the relevance of rule restrictions. We then study the generative capacity of pure CCG in Section 3; this section also presents our main result. In Section 4, we show that this result still holds for multi-modal CCG. Section 5 concludes the paper with a discussion of the relevance of our findings.

2 Combinatory Categorial Grammar

We start by providing formal definitions for categories, syntactic rules, and grammars, and then discuss the relevance of rule restrictions for CCG.

2.1 Categories

Given a finite set A of *atomic categories*, the *set of categories over* A is the smallest set C such that $A \subseteq C$, and $(\mathsf{x}/\mathsf{y}), (\mathsf{x}\backslash\mathsf{y}) \in C$ whenever $\mathsf{x}, \mathsf{y} \in C$. A category x/y represents a function that seeks a string with category y to the right (indicated by the forward slash) and returns a new string with category x; a category $\mathsf{x}\backslash\mathsf{y}$ instead seeks its argument to the left (indicated by the backward slash). In the remainder of this paper, we use lowercase sans-serif letters such as $\mathsf{x}, \mathsf{y}, \mathsf{z}$ as variables for categories, and the vertical bar $|$ as a variable for slashes. In order to save some parentheses, we understand slashes as left-associative operators, and write a category such as $(\mathsf{x}/\mathsf{y})\backslash\mathsf{z}$ as $\mathsf{x}/\mathsf{y}\backslash\mathsf{z}$.

The list of *arguments* of a category c is defined recursively as follows: If c is atomic, then it has no arguments. If $\mathsf{c} = \mathsf{x}|\mathsf{y}$ for some categories x and y, then the arguments of c are the slashed category $|\mathsf{y}$, plus the arguments of x. We number the arguments of a category from outermost to innermost. The *arity* of a category is the number of its arguments. The *target* of a category c is the atomic category that remains when stripping c of its arguments.

x/y	y	$\Rightarrow$	x	forward application $>$
y	$\mathsf{x}\backslash\mathsf{y}$	$\Rightarrow$	x	backward application $<$
x/y	y/z	$\Rightarrow$	x/z	forward harmonic composition $>\mathsf{B}$
$\mathsf{y}\backslash\mathsf{z}$	$\mathsf{x}\backslash\mathsf{y}$	$\Rightarrow$	$\mathsf{x}\backslash\mathsf{z}$	backward harmonic composition $<\mathsf{B}$
x/y	$\mathsf{y}\backslash\mathsf{z}$	$\Rightarrow$	$\mathsf{x}\backslash\mathsf{z}$	forward crossed composition $>\mathsf{B}_\times$
y/z	$\mathsf{x}\backslash\mathsf{y}$	$\Rightarrow$	x/z	backward crossed composition $<\mathsf{B}_\times$

Figure 1: The core set of rules of CCG.

2.2 Rules

The syntactic rules of CCG are directed versions of combinators in the sense of combinatory logic (Curry et al., 1958). Figure 1 lists a core set of commonly assumed rules, derived from functional application and the B combinator, which models functional composition. When talking about these rules, we refer to the premise containing the argument $|\mathsf{y}$ as the *primary premise*, and to the other premise as the *secondary premise* of the rule.

The rules in Figure 1 can be generalized into composition rules of higher degrees. These are defined as follows, where $n \geq 0$ and β is a variable for a sequence of n arguments.

x/y	$\mathsf{y}\beta$	$\Rightarrow$	$\mathsf{x}\beta$	generalized forward composition $>^n$
$\mathsf{y}\beta$	$\mathsf{x}\backslash\mathsf{y}$	$\Rightarrow$	$\mathsf{x}\beta$	generalized backward composition $<^n$

We call the value n the *degree* of the composition rule. Note that the rules in Figure 1 are the special cases for $n = 0$ and $n = 1$.

Apart from the core rules given in Figure 1, some versions of CCG also use rules derived from the S and T combinators of combinatory logic, called *substitution* and *type-raising*, the latter restricted to the lexicon. However, since our main point of reference in this paper, the CCG formalism defined by Vijay-Shanker and Weir (1994), does *not* use such rules, we will not consider them here, either.

2.3 Grammars and Derivations

With the set of rules in place, we can define a *pure combinatory categorial grammar* (PCCG) as a construct $G = (A, \Sigma, L, \mathsf{s})$, where A is an alphabet of *atomic categories*, $\mathsf{s} \in A$ is a distinguished atomic category called the *final category*, Σ is a finite set of *terminal symbols*, and L is a finite relation between symbols in Σ and categories over A, called the *lexicon*. The elements of the lexicon L are called *lexicon entries*, and we represent them using the notation $\sigma \vdash \mathsf{x}$, where $\sigma \in \Sigma$ and x is a category over A. A category that occurs in a lexicon entry is called a *lexical category*.

A derivation in a grammar G can be represented as a *derivation tree* as follows. Given a string $w \in \Sigma^*$, we choose a lexicon entry for each occurrence of a symbol in w, line up the respective lexical categories from left to right, and apply admissible rules to adjacent pairs of categories. After the application of a rule, only the conclusion is available for future applications. We iterate this process until we end up with a single category. The string w is called the *yield* of the resulting derivation tree. A derivation tree is *complete*, if the last category is the final category of G. The *language generated by* G, denoted by $L(G)$, is formed by the yields of all complete derivation trees.

2.4 Degree Restrictions

Work on CCG generally assumes an upper bound on the degree of composition rules that can be used in derivations. We also employ this restriction, and only consider grammars with compositions of some bounded (but arbitrary) degree $n \geq 0$.[1] CCG with unbounded-degree compositions is more expressive than bounded-degree CCG or TAG (Weir and Joshi, 1988).

Bounded-degree grammars have a number of useful properties, one of which we mention here. The following lemma rephrases Lemma 3.1 in Vijay-Shanker and Weir (1994).

Lemma 1 *For every grammar G, every argument in a derivation of G is the argument of some lexical category of G.*

As a consequence, there is only a finite number of categories that can occur as arguments in some derivation. In the presence of a bound on the degree of composition rules, this implies the following:

Lemma 2 *For every grammar G, there is a finite number of categories that can occur as secondary premises in derivations of G.*

Proof. The arity of a secondary premise c can be written as $m + n$, where m is the arity of the first argument of the corresponding primary premise, and n is the degree of the rule applied. Since each argument is an argument of some lexical category of G (Lemma 1), and since n is assumed to be bounded, both m and n are bounded. Hence, there is a bound on the number of choices for c. ∎

Note that the number of categories that can occur as *primary* premises is generally unbounded even in a grammar with bounded degree.

[1] For practical grammars, $n \leq 4$.

2.5 Rule Restrictions

The rule set of pure CCG is universal: the difference between the grammars of different languages should be restricted to different choices of categories in the lexicon. This is what makes pure CCG a lexicalized grammar formalism (Steedman and Baldridge, 2010). However, most practical CCG grammars rely on the possibility to exclude or restrict certain rules. For example, Steedman (2001) bans the rule of forward crossed composition from his grammar of English, and stipulates that the rule of backward crossed composition may be applied only if both of its premises share the common target category s, representing sentences. Exclusions and restrictions of rules are also assumed in much of the language-theoretic work on CCG. In particular, they are essential for the formalism used in the aforementioned equivalence proof for CCG and TAG (Vijay-Shanker and Weir, 1994).

To illustrate the formal relevance of rule restrictions, suppose that we wanted to write a pure CCG that generates the language

$$L_3 \;=\; \{\, a^n b^n c^n \mid n \geq 1 \,\},$$

which is not context-free. An attempt could be

$$G_1 \;=\; (\{\mathsf{s},\mathsf{a},\mathsf{b},\mathsf{c}\},\{a,b,c\},L,\mathsf{s}),$$

where the lexicon L is given as follows:

$$a \vdash \mathsf{a}\,,\; b \vdash \mathsf{s/c\backslash a}\,,\; b \vdash \mathsf{b/c\backslash a}\,,$$
$$b \vdash \mathsf{s/c/b\backslash a}\,,\; b \vdash \mathsf{s/c/b\backslash a}\,,\; c \vdash \mathsf{c}\,.$$

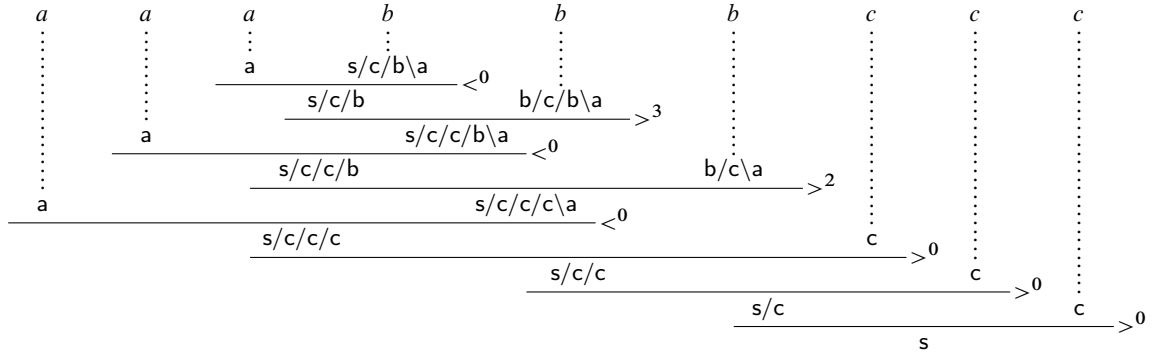
From a few sample derivations like the one given in Figure 2a, we can convince ourselves that G_1 generates all strings of the form $a^n b^n c^n$, for any $n \geq 1$. However, a closer inspection reveals that it also generates other, unwanted strings—in particular, strings of the form $(ab)^n c^n$, as witnessed by the derivation given in Figure 2b.

Now suppose that we would have a way to only allow those instances of generalized composition in which the secondary premise has the form $\mathsf{b/c/b\backslash a}$ or $\mathsf{b/c\backslash a}$. Then the compositions

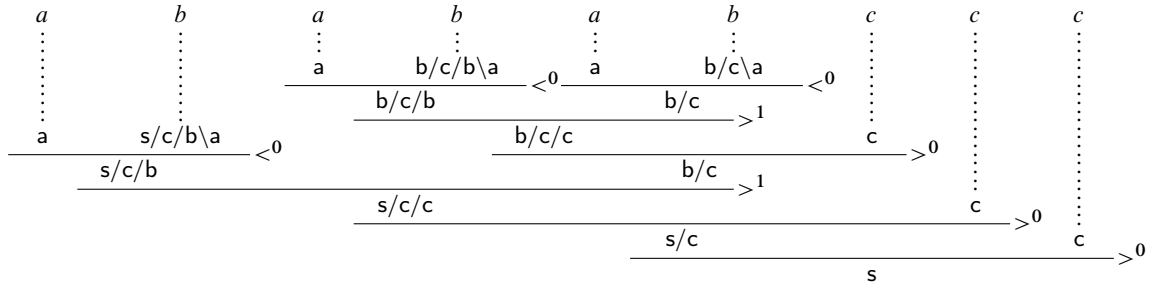
$$\frac{\mathsf{b/c/b}\quad\mathsf{b/c}}{\mathsf{b/c/c}}\,{>}^1 \quad\text{and}\quad \frac{\mathsf{s/c/b}\quad\mathsf{b/c}}{\mathsf{s/c/c}}\,{>}^1$$

would be disallowed, and it is not hard to see that G_1 would generate exactly $a^n b^n c^n$.

As we will show in this paper, our attempt to capture L_3 with a pure CCG grammar failed not only because we could not think of one: L_3 cannot be generated by *any* pure CCG.



(a) Derivation of the string $aaabbbccc$.



(b) Derivation of the string $abababccc$.

Figure 2: Two derivations of the grammar G_1.

3 The Generative Capacity of Pure CCG

We will now develop a formal argument showing that rule restrictions increase the weak generative capacity of CCG. We will first prove that pure CCG is still more expressive than context-free grammar. We will then spend the rest of this section working towards the result that pure CCG is strictly less expressive than CCG with rule restrictions. Our main technical result will be the following:

Theorem 1 *Every language that can be generated by a pure CCG has a Parikh-equivalent context-free sublanguage.*

Here, two languages L and L' are called *Parikh-equivalent* if every string in L is the permutation of a string in L' and vice versa.

3.1 CFG $\subsetneq$ PCCG

Proposition 1 *The class of languages generated by pure CCG properly includes the class of context-free languages.*

Proof. To see the inclusion, it suffices to note that pure CCG when restricted to application rules is the same as *AB-grammar*, the classical categorial formalism investigated by Ajdukiewicz and Bar-Hillel (Bar-Hillel et al., 1964). This formalism is weakly equivalent to context-free grammar.

To see that the inclusion is proper, we can go back to the grammar G_1 that we gave in Section 2.5. We have already discussed that the language L_3 is included in $L(G_1)$. We can also convince ourselves that all strings generated by the grammar G_1 have an equal number of as, bs and cs. Consider now the regular language $R = a^*b^*c^*$. From our observations, it follows that $L(G_1) \cap R = L_3$. Since context-free languages are closed under intersection with regular languages, we find that $L(G_1)$ can be context-free only if L_3 is. Since L_3 is not context-free, we therefore conclude that $L(G_1)$ is not context-free, either. ∎

Two things are worth noting. First, our result shows that the ability of CCG to generate non-context-free languages does not hinge on the availability of substitution and type-raising rules: The derivations of G_1 only use generalized compositions. Neither does it require the use of functional argument categories: The grammar G_1 is *first-order* in the sense of Koller and Kuhlmann (2009).

Second, it is important to note that if the composition degree n is restricted to 0 or 1, pure CCG actually collapses to context-free expressive power. This is clear for $n = 0$ because of the equivalence to AB grammar. For $n = 1$, observe that the arity of the result of a composition is at most as high as

that of each premise. This means that the arity of any derived category is bounded by the maximal arity of lexical categories in the grammar, which together with Lemma 1 implies that there is only a finite set of derivable categories. The set of all valid derivations can then be simulated by a context-free grammar. In the presence of rules with $n \geq 2$, the arities of derived categories can grow unboundedly.

3.2 Active and Inactive Arguments

In the remainder of this section, we will develop the proof of Theorem 1, and use it to show that the generative capacity of PCCG is strictly smaller than that of CCG with rule restrictions. For the proof, we adopt a certain way to view the information flow in CCG derivations. Consider the following instance of forward harmonic composition:

$$\mathsf{a/b}\ \ \mathsf{b/c}\ \Rightarrow\ \mathsf{a/c}$$

This rule should be understood as obtaining its conclusion $\mathsf{a/c}$ from the primary premise $\mathsf{a/b}$ by the removal of the argument $\mathsf{/b}$ and the subsequent transfer of the argument $\mathsf{/c}$ from the secondary premise. With this picture in mind, we will view the two occurrences of $\mathsf{/c}$ in the secondary premise and in the conclusion as two occurrences of one and the same argument. Under this perspective, in a given derivation, an argument has a lifespan that starts in a lexical category and ends in one of two ways: either in the primary or in the secondary premise of a composition rule. If it ends in a primary premise, it is because it is matched against a subcategory of the corresponding secondary premise; this is the case for the argument $\mathsf{/b}$ in the example above. We will refer to such arguments as *active*. If an argument ends its life in a secondary premise, it is because it is consumed as part of a higher-order argument. This is the case for the argument $\mathsf{/c}$ in the secondary premise of the following rule instance:

$$\mathsf{a/(b/c)}\ \ \mathsf{b/c/d}\ \Rightarrow\ \mathsf{a/d}$$

(Recall that we assume that slashes are left-associative.) We will refer to such arguments as *inactive*. Note that the status of an argument as either active or inactive is not determined by the grammar, but depends on a concrete derivation.

The following lemma states an elementary property in connection with active and inactive arguments, which we will refer to as *segmentation*:

Lemma 3 *Every category that occurs in a CCG derivation has the general form* $\mathsf{a}\alpha\beta$*, where* a *is an atomic category,* α *is a sequence of inactive arguments, and* β *is a sequence of active arguments.*

Proof. The proof is by induction on the depth of a node in the derivation. The property holds for the root (which is labeled with the final category), and is transferred from conclusions to premises. ∎

3.3 Transformation

The fundamental reason for why the example grammar G_1 from Section 2.5 overgenerates is that in the absence of rule restrictions, we have no means to control the point in a derivation at which a category combines with its arguments. Consider the examples in Figure 2: It is because we cannot ensure that the bs finish combining with the other bs before combining with the cs that the undesirable word order in Figure 2b has a derivation. To put it as a slogan: Permuting the words allows us to saturate arguments prematurely.

In this section, we show that this property applies to all pure CCGs. More specifically, we show that, in a derivation of a pure CCG, almost all active arguments of a category can be saturated before that category is used as a secondary premise; at most one active argument must be transferred to the conclusion of that premise. Conversely, any derivation that still contains a category with at least two active arguments can be transformed into a new derivation that brings us closer to the special property just characterized.

We formalize this transformation by means of a system of rewriting rules in the sense of Baader and Nipkow (1998). The rules are given in Figure 3. To see how they work, let us consider the first rule, R1; the other ones are symmetric. This rules states that, whenever we see a derivation in which a category of the form $\mathsf{x/y}$ (here marked as **A**) is combined with a category of the form $\mathsf{y}\beta\mathsf{/z}$ (marked as **B**), and the result of this combination is combined with a category of the form $\mathsf{z}\gamma$ (**C**), then the resulting category can also be obtained by 'rotating' the derivation to first saturate $\mathsf{/z}$ by combining **B** with **C**, and only then do the combination with **A**. When applying these rotations exhaustively, we end up with a derivation in which almost all active arguments of a category are saturated before that category is used as a secondary premise. Applying the transformation to the derivation in Figure 2a, for instance, yields the derivation in Figure 2b.

We need the following result for some of the lemmas we prove below. We call a node in a deriv-

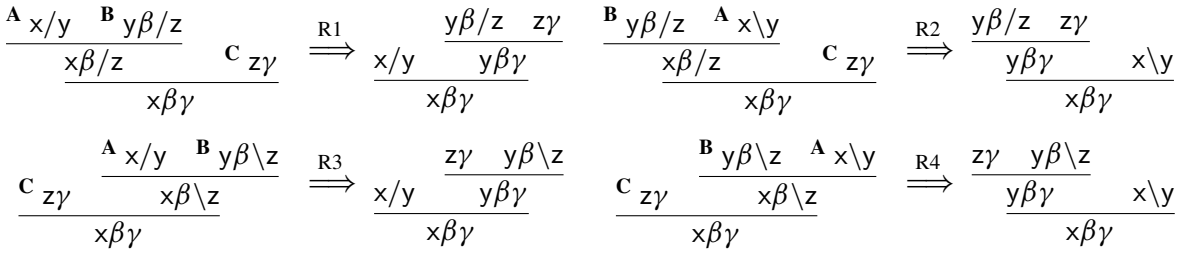


Figure 3: Rewriting rules used in the transformation. Here, γ represents a (possibly empty) sequence of arguments, and β represents a sequence of arguments in which the first (outermost) argument is active.

ation *critical* if its corresponding category contains more than one active argument and it is the secondary premise of a rule. We say that u is a *highest* critical node if there is no other critical node whose distance to the root is shorter.

Lemma 4 *If u is a highest critical node, then we can apply one of the transformation rules to the grandparent of u.*

Proof. Suppose that the category at u has the form $\mathsf{y}\beta/\mathsf{z}$, where $/\mathsf{z}$ is an active argument, and the first argument in β is active as well. (The other possible case, in which the relevant occurrence has the form $\mathsf{y}\beta\backslash\mathsf{z}$, can be treated symmetrically.) Since u is a secondary premise, it is involved in an inference of one of the following two forms:

$$\frac{\mathsf{x}/\mathsf{y} \quad \mathsf{y}\beta/\mathsf{z}}{\mathsf{x}\beta/\mathsf{z}} \qquad \frac{\mathsf{y}\beta/\mathsf{z} \quad \mathsf{x}\backslash\mathsf{y}}{\mathsf{x}\beta/\mathsf{z}}$$

Since u is a highest critical node, the conclusion of this inference is not a critical node itself; in particular, it is not a secondary premise. Therefore, the above inferences can be extended as follows:

$$\frac{\dfrac{\mathsf{x}/\mathsf{y} \quad \mathsf{y}\beta/\mathsf{z}}{\mathsf{x}\beta/\mathsf{z}} \quad \mathsf{z}\gamma}{\mathsf{x}\beta\gamma} \qquad \frac{\dfrac{\mathsf{y}\beta/\mathsf{z} \quad \mathsf{x}\backslash\mathsf{y}}{\mathsf{x}\beta/\mathsf{z}} \quad \mathsf{z}\gamma}{\mathsf{x}\beta\gamma}$$

These partial derivations match the left-hand side of the rewriting rules R1 and R2, respectively. Hence, we can apply a rewriting rule to the derivation. ∎

We now show that the transformation is well-defined, in the sense that it terminates and transforms derivations of a grammar G into new derivations of G.

Lemma 5 *The rewriting of a derivation tree ends after a finite number of steps.*

Proof. We assign natural numbers to the nodes of a derivation tree as follows. Each leaf node is assigned the number 0. For an inner node u, which corresponds to the conclusion of a composition rule, let m, n be the numbers assigned to the nodes corresponding to the primary and secondary premise, respectively. Then u is assigned the number $1 + 2m + n$. Suppose now that we have associated premise **A** with the number x, premise **B** with the number y, and premise **C** with the number z. It is then easy to verify that the conclusion of the partial derivation on the left-hand side of each rule has the value $3 + 4x + 2y + z$, while the conclusion of the right-hand side has the value $2 + 2x + 2y + z$. Thus, each step decreases the value of a derivation tree under our assignment by the amount $1 + 2x$. Since this value is positive for all choices of x, the rewriting ends after a finite number of steps. ∎

To convince ourselves that our transformation does not create ill-formed derivations, we need to show that none of the rewriting rules necessitates the use of composition operations whose degree is higher than the degree of the operations used in the original derivation.

Lemma 6 *Applying the rewriting rules from the top down does not increase the degree of the composition operations.*

Proof. The first composition rule used in the left-hand side of each rewriting rule has degree $|\beta| + 1$, the second rule has degree $|\gamma|$; the first rule used in the right-hand side has degree $|\gamma|$, the second rule has degree $|\beta| + |\gamma|$. To prove the claim, it suffices to show that $|\gamma| \leq 1$. This is a consequence of the following two observations.

1. In the category $\mathsf{x}\beta\gamma$, the arguments in γ occur on top of the arguments in β, the first of which is active. Using the segmentation property stated in Lemma 3, we can therefore infer that γ does not contain any inactive arguments.

2. Because we apply rules top-down, premise **B** is a highest critical node in the derivation (by Lemma 4). This means that the category at premise **C** contains at most one active argument; otherwise, premise **C** would be a critical node closer to the root than premise **B**. ∎

We conclude that, if we rewrite a derivation d of G top-down until exhaustion, then we obtain a new valid derivation d'. We call all derivations d' that we can build in this way *transformed*. It is easy to see that a derivation is transformed if and only if it contains no critical nodes.

3.4 Properties of Transformed Derivations

The special property established by our transformation has consequences for the generative capacity of pure CCG. In particular, we will now show that the set of all transformed derivations of a given grammar yields a context-free language. The crucial lemma is the following:

Lemma 7 *For every grammar G, there is some $k \geq 0$ such that no category in a transformed derivation of G has arity greater than k.*

Proof. The number of inactive arguments in the primary premise of a rule does not exceed the number of inactive arguments in the conclusion. In a transformed derivation, a symmetric property holds for active arguments: Since each secondary premise contains at most one active argument, the number of active arguments in the conclusion of a rule is not greater than the number of active arguments in its primary premise. Taken together, this implies that the arity of a category that occurs in a transformed derivation is bounded by the sum of the maximal arity of a lexical category (which bounds the number of active arguments), and the maximal arity of a secondary premise (which bounds the number of inactive arguments). Both of these values are bounded in G. ∎

Lemma 8 *The yields corresponding to the set of all transformed derivations of a pure CCG form a context-free language.*

Proof. Let G be a pure CCG. We construct a context-free grammar G_T that generates the yields of the set of all transformed derivations of G.

As the set of terminals of G_T, we use the set of terminals of G. To form the set of nonterminals, we take all categories that can occur in a transformed derivation of G, and mark each argument as either 'active' ($+$) or 'inactive' ($-$), in all possible ways that respect the segmentation property stated in Lemma 3. Note that, because of Lemma 7 and Lemma 1, the set of nonterminals is finite. As the start symbol, we use s, the final category of G.

The set of productions of G_T is constructed as follows. For each lexicon entry $\sigma \vdash \mathsf{c}$ of G, we include all productions of the form $x \to \sigma$, where x is some marked version of c. These productions represent all valid guesses about the activity of the arguments of c during a derivation of G. The remaining productions encode all valid instantiations of composition rules, keeping track of active and inactive arguments to prevent derivations with critical nodes. More specifically, they have the form

$$\mathsf{x}\beta \to \mathsf{x}/\mathsf{y}^+\ \mathsf{y}\beta \quad \text{or} \quad \mathsf{x}\beta \to \mathsf{y}\beta\ \mathsf{x}\backslash\mathsf{y}^+\,,$$

where the arguments in the y-part of the secondary premise are all marked as inactive, the sequence β contains at most one argument marked as active, and the annotations of the left-hand side nonterminal are copied over from the corresponding annotations on the right-hand side.

The correctness of the construction of G_T can be proved by induction on the length of a transformed derivation of G on the one hand, and the length of a derivation of G_T on the other hand. ∎

3.5 PCCG $\subsetneq$ CCG

We are now ready to prove our main result, repeated here for convenience.

Theorem 1 *Every language that can be generated by a pure CCG grammar has a Parikh-equivalent context-free sublanguage.*

Proof. Let G be a pure CCG, and let L_T be the set of yields of the transformed derivations of G. Inspecting the rewriting rules, it is clear that every string of $L(G)$ is the permutation of a string in L_T: the transformation only rearranges the yields. By Lemma 8, we also know that L_T is context-free. Since every transformed derivation is a valid derivation of G, we have $L_T \subseteq L(G)$. ∎

As an immediate consequence, we find:

Proposition 2 *The class of languages generated by pure CCG cannot generate all languages that can be generated by CCG with rule restrictions.*

Proof. The CCG formalism considered by Vijay-Shanker and Weir (1994) can generate the non-context-free language L_3. However, the only Parikh-equivalent sublanguage of that language is L_3 itself. From Theorem 1, we therefore conclude that L_3 cannot be generated by pure CCG. ∎

In the light of the equivalence result established by Vijay-Shanker and Weir (1994), this means that pure CCG cannot generate all languages that can be generated by TAG.

4 Multi-Modal CCG

We now extend Theorem 1 to multi-modal CCG. We will see that at least for a popular version of multi-modal CCG, the B&K-CCG formalism presented by Baldridge and Kruijff (2003), the proof can be adapted quite straightforwardly. This means that even B&K-CCG becomes less expressive when rule restrictions are disallowed.

4.1 Multi-Modal CCG

The term 'multi-modal CCG' (MM-CCG) refers to a family of extensions to CCG which attempt to bring some of the expressive power of Categorial Type Logic (Moortgat, 1997) into CCG. Slashes in MM-CCG have *slash types*, and rules can be restricted to only apply to arguments that have slashes of the correct type. The idea behind this extension is that many constraints that in ordinary CCG can only be expressed in terms of rule restrictions can now be specified in the lexicon entries by giving the slashes the appropriate types.

The most widely-known version of multi-modal CCG is the formalism defined by Baldridge and Kruijff (2003) and used by Steedman and Baldridge (2010); we refer to it as B&K-CCG. This formalism uses an inventory of four slash types, $\{\star, \times, \diamond, \cdot\}$, arranged in a simple type hierarchy: $\star$ is the most general type, $\cdot$ the most specific, and $\times$ and $\diamond$ are in between. Every slash in a B&K-CCG lexicon is annotated with one of these slash types.

The combinatory rules in B&K-CCG, given in Figure 4, are defined to be sensitive to the slash types. In particular, slashes with the types $\diamond$ and $\times$ can only be eliminated by harmonic and crossed compositions, respectively.[2] Thus, a grammar writer can constrain the application of harmonic and crossed composition rules to certain categories by assigning appropriate types to the slashes of this category in the lexicon. Application rules apply to slashes of any type. As before, we call an MM-CCG grammar *pure* if it only uses application and generalized compositions, and does not provide means to restrict rule applications.

[2] Our definitions of generalized harmonic and crossed composition are the same as the ones used by Hockenmaier and Young (2008), but see the discussion in Section 4.3.

$x/_\star y \;\; y \Rightarrow x$		forward application
$y \;\; x\backslash_\star y \Rightarrow x$		backward application
$x/_\diamond y \;\; y/_\diamond z\beta \Rightarrow x/_\diamond z\beta$		forward harmonic composition
$x/_\times y \;\; y\backslash_\times z\beta \Rightarrow x\backslash_\times z\beta$		forward crossed composition
$y\backslash_\diamond z\beta \;\; x\backslash_\diamond y \Rightarrow x\backslash_\diamond z\beta$		backward harmonic composition
$y/_\times z\beta \;\; x\backslash_\times y \Rightarrow x/_\times z\beta$		backward crossed composition

Figure 4: Rules in B&K-CCG.

4.2 Rule Restrictions in B&K-CCG

We will now see what happens to the proof of Theorem 1 in the context of pure B&K-CCG. There is only one point in the entire proof that could be damaged by the introduction of slash types, and that is the result that if a transformation rule from Figure 3 is applied to a correct derivation, then the result is also grammatical. For this, it must not only be the case that the degree on the composition operations is preserved (Lemma 6), but also that the transformed derivation remains consistent with the slash types. Slash types make the derivation process sensitive to word order by restricting the use of compositions to categories with the appropriate type, and the transformation rules permute the order of the words in the string. There is a chance therefore that a transformed derivation might not be grammatical in B&K-CCG.

We now show that this does not actually happen, for rule R3; the other three rules are analogous. Using s_1, s_2, s_3 as variables for the relevant slash types, rule R3 appears in B&K-CCG as follows:

$$\dfrac{z\gamma \quad \dfrac{x/_{s_1} y \quad y|_{s_2} w\beta\backslash_{s_3} z}{x|_{s_2} w\beta\backslash_{s_3} z}}{x|_{s_2} w\beta\gamma} \overset{\text{R3}}{\Longrightarrow} \dfrac{x/_{s_1} y \quad \dfrac{z\gamma \quad y|_{s_2} w\beta\backslash_{s_3} z}{y|_{s_2} w\beta\gamma}}{x|_{s_2} w\beta\gamma}$$

Because the original derivation is correct, we know that, if the slash of w is forward, then s_1 and s_2 are subtypes of $\diamond$; if the slash is backward, they are subtypes of $\times$. A similar condition holds for s_3 and the first slash in γ; if γ is empty, then s_3 can be anything because the second rule is an application.

After the transformation, the argument $/_{s_1} y$ is used to compose with $y|_{s_2} w\beta\gamma$. The direction of the slash in front of the w is the same as before, so the (harmonic or crossed) composition is still compatible with the slash types s_1 and s_2. An analogous argument shows that the correctness of combining $\backslash_{s_3} z$ with γ carries over from the left to the right-hand side. Thus the transformation maps grammatical derivations into grammatical derivations. The rest of the proof in Section 3 continues to work literally, so we have the following result:

Theorem 2 *Every language that can be generated by a pure B&K-CCG grammar contains a Parikh-equivalent context-free sublanguage.*

This means that pure B&K-CCG is just as unable to generate L_3 as pure CCG is. In other words, the weak generative capacity of CCG with rule restrictions, and in particular that of the formalism considered by Vijay-Shanker and Weir (1994), is strictly greater than the generative capacity of pure B&K-CCG—although we conjecture (but cannot prove) that pure B&K-CCG is still more expressive than pure non-modal CCG.

4.3 Towards More Expressive MM-CCGs

To put the result of Theorem 2 into perspective, we will now briefly consider ways in which B&K-CCG might be modified in order to obtain a pure multi-modal CCG that is weakly equivalent to CCG in the style of Vijay-Shanker and Weir (1994). Such a modification would have to break the proof in Section 4.2, which is harder than it may seem at first glance. For instance, simply assuming a more complex type system will not do it, because the arguments $\backslash_{s_3}\mathsf{z}$ and $/_{s_1}\mathsf{y}$ are eliminated using the same rules in the original and the transformed derivations, so if the derivation step was valid before, it will still be valid after the transformation. Instead, we believe that it is necessary to make the composition rules sensitive to the categories inside β and γ instead of only the arguments $\backslash_{s_3}\mathsf{z}$ and $/_{s_1}\mathsf{y}$, and we can see two ways how to do this.

First, one could imagine a version of multi-modal CCG with unary modalities that can be used to mark certain category occurrences. In such an MM-CCG, the composition rules for a certain slash type could be made sensitive to the presence or absence of unary modalities in β. Say for instance that the slash type s_1 in the modalized version of R3 in Section 4.2 would require that no category in the secondary argument is marked with the unary modality '$\Box$', but β contains a category marked with '$\Box$'. Then the transformed derivation would be ungrammatical.

A second approach concerns the precise definition of the generalized composition rules, about which there is a surprising degree of disagreement. We have followed Hockenmaier and Young (2008) in classifying instances of generalized forward composition as harmonic if the innermost slash of the secondary argument is forward and crossed if it is backward. However, generalized forward composition is sometimes only accepted as harmonic if *all* slashes of the secondary argument are forward (see e.g. Baldridge (2002) (40, 41), Steedman (2001) (19)). At the same time, based on the principle that CCG rules should be derived from proofs of Categorial Type Logic as Baldridge (2002) does, it can be argued that generalized composition rules of the form $\mathsf{x/y}\ \ \mathsf{y/z\backslash w} \Rightarrow \mathsf{x/z\backslash w}$, which we have considered as harmonic, should actually be classified as crossed, due to the presence of a slash of opposite directionality in front of the w. This definition would break our proof. Thus our result might motivate further research on the 'correct' definition of generalized composition rules, which might then strengthen the generative capacity of pure MM-CCG.

5 Conclusion

In this paper, we have shown that the weak generative capacity of pure CCG and even pure B&K-CCG crucially depends on the ability to restrict the application of individual rules. This means that these formalisms cannot be fully lexicalized, in the sense that certain languages can only be described by selecting language-specific rules.

Our result generalizes Koller and Kuhlmann's (2009) result for pure *first-order* CCG. Our proof is not as different as it looks at first glance, as their construction of mapping a CCG derivation to a valency tree and back to a derivation provides a different transformation on derivation trees. Our transformation is also technically related to the normal form construction for CCG parsing presented by Eisner (1996).

Of course, at the end of the day, the issue that is more relevant to computational linguistics than a formalism's ability to generate artificial languages such as L_3 is how useful it is for modeling *natural* languages. CCG, and multi-modal CCG in particular, has a very good track record for this. In this sense, our formal result can also be understood as a contribution to a discussion about the expressive power that is needed to model natural languages.

Acknowledgments

We have profited enormously from discussions with Jason Baldridge and Mark Steedman, and would also like to thank the anonymous reviewers for their detailed comments.

References

Franz Baader and Tobias Nipkow. 1998. *Term Rewriting and All That*. Cambridge University Press.

Jason Baldridge and Geert-Jan M. Kruijff. 2003. Multi-modal Combinatory Categorial Grammar. In *Proceedings of the Tenth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 211–218, Budapest, Hungary.

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

Yehoshua Bar-Hillel, Haim Gaifman, and Eli Shamir. 1964. On categorial and phrase structure grammars. In *Language and Information: Selected Essays on their Theory and Application*, pages 99–115. Addison-Wesley.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 176–182, Geneva, Switzerland.

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

Haskell B. Curry, Robert Feys, and William Craig. 1958. *Combinatory Logic. Volume 1*. Studies in Logic and the Foundations of Mathematics. North-Holland.

Jason Eisner. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 79–86, Santa Cruz, CA, USA.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 335–342, Philadelphia, USA.

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

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoining Grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–123. Springer.

Alexander Koller and Marco Kuhlmann. 2009. Dependency trees and the strong generative capacity of CCG. In *Proceedings of the Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 460–468, Athens, Greece.

Michael Moortgat. 1997. Categorial type logics. In *Handbook of Logic and Language*, chapter 2, pages 93–177. Elsevier.

David Reitter, Julia Hockenmaier, and Frank Keller. 2006. Priming effects in combinatory categorial grammar. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 308–316, Sydney, Australia.

Mark Steedman and Jason Baldridge. 2010. Combinatory categorial grammar. In R. Borsley and K. Borjars, editors, *Non-Transformational Syntax*. Blackwell. Draft 7.0, to appear.

Mark Steedman. 2001. *The Syntactic Process*. MIT Press.

K. Vijay-Shanker and David J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.

David J. Weir and Aravind K. Joshi. 1988. Combinatory categorial grammars: Generative power and relationship to linear context-free rewriting systems. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 278–285, Buffalo, NY, USA.