Non-local scrambling: the equivalence of TAG and CCG revisited

Julia Hockenmaier and Peter Young
Department of Computer Science, University of Illinois,
201 N. Goodwin Ave., Urbana-Champaign, 61801 IL, USA
{juliahmr,pyoung2}@cs.uiuc.edu

Abstract

It is well known that standard TAG cannot deal with certain instances of long-distance scrambling in German (Rambow, 1994). That CCG can deal with many instances of non-local scrambling in languages such as Turkish has previously been observed (e.g. by Hoffman (1995a) and Baldridge (2002)). We show here that CCG can derive German scrambling cases which are problematic for TAG, and give CCG analyses for other German constructions that require more expressive power than TAG provides. Such analyses raise the question of the linguistic significance of the TAG-CCG equivalence. We revisit the original equivalence proof, and show that a careful examination of the translation of CCG and TAG into Indexed Grammar reveals that the IG which is strongly equivalent to CCG can generate dependencies which the corresponding IG obtained from an LTAG cannot generate.

1 Introduction

Vijay-Shanker and Weir (1994) proved that Tree-Adjoining Grammar (TAG (Joshi and Schabes, 1997)), Combinatory Categorial Grammar (CCG (Steedman, 2000)) and Linear Indexed Grammars (LIG, (Gazdar, 1988)) are weakly equivalent, i.e. can generate the same sets of strings. All of these grammars can generate the languages $\{a^n b^n c^n d^n\}$ (which does not correspond to any known construction in natural language), and $\{a^n b^n\}$ with cross-serial dependencies (i.e. $a_1 ... a_n b_1 ... b_n$), corresponding to the cross-serial dependencies that arise in Dutch (Bresnan et al., 1982) and Swiss German (Shieber, 1985).

Although this result has important algorithmic consequences (Vijay-Shanker and Weir, 1993), it is easy to overestimate its linguistic relevance. Weak equivalence does, of course, not necessarily imply that two formalisms are capable of recovering the same set of dependencies between the elements of a string. Since the notion of strong equivalence is often hard to define, strong equivalene proofs are rarely found in the literature. But examples of structures that can only be analyzed in one formalism can provide insight into where their strong generative capacities differ.

2 Combinatory Categorial Grammar

In addition to function application ($>$ and $<$), CCG allows the combinatory rules of (generalized) function composition ($\mathbf{B}_n$), which allows a functor $\mathsf{X|Y}$ to compose with another functor $\mathsf{Y|Z}_1 ... \mathsf{Z}_n$ to form a category $\mathsf{X|Z}_1 ... \mathsf{Z}_n$, and type-raising $\mathbf{T}$, which allows a category X to be transformed into a category $\mathsf{T/(T\backslash X)}$ or $\mathsf{T\backslash(T/X)}$:

$\mathsf{X/Y}$	Y	$\Rightarrow_{>}$	X
Y	$\mathsf{X\backslash Y}$	$\Rightarrow_{<}$	X
$\mathsf{X/Y}$	$\mathsf{Y/Z}_1 \mid ... \mathsf{Z}_n$	$\Rightarrow_{>\mathbf{B}^n}$	$\mathsf{X/Z}_1 ... \mathsf{Z}_n$
$\mathsf{Y\backslash Z}_1 ... \mathsf{Z}_n$	$\mathsf{X\backslash Y}$	$\Rightarrow_{<\mathbf{B}^n}$	$\mathsf{X\backslash Z}_1 .... \mathsf{Z}_n$
$\mathsf{X/Y}$	$\mathsf{Y\backslash Z}_1 ... \mathsf{Z}_n$	$\Rightarrow_{>\mathbf{B}_\times{}^n}$	$\mathsf{X\backslash Z}_1 .. \mathsf{Z}_n$
$\mathsf{Y/Z}_1 ... \mathsf{Z}_n$	$\mathsf{X\backslash Y}$	$\Rightarrow_{<\mathbf{B}_\times{}^n}$	$\mathsf{X/Z}_1 .. \mathsf{Z}_n$
X		$\Rightarrow_{>\mathbf{T}}$	$\mathsf{T/(T\backslash X)}$
X		$\Rightarrow_{<\mathbf{T}}$	$\mathsf{T\backslash(T/X)}$

Steedman (2000) furthermore uses a unary topicalization rule, which is only allowed to be applied to a sentence-initial constituent:

$$\mathsf{X} \quad \Rightarrow_{>\mathbf{T}} \quad \mathsf{T/(T/X)}$$

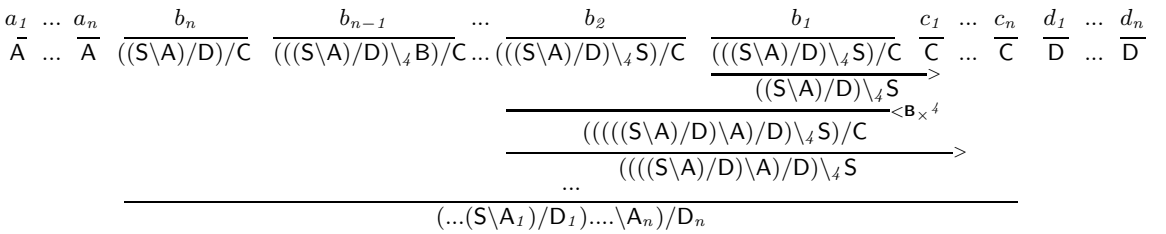


Figure 1: Type-raising is not required to derive $a^n b^n c^n d^n$ in CCG.

Both the maximal arity n up to which generalized composition $\mathbf{B}_n$ is allowed and the maximal arity k of the variable T that results from type-raising are assumed to be bounded (typically to the maximal arity of lexical categories required by a language (Steedman, 2000)). These bounds are known to be important: Weir (1988) shows that if there is no bound on generalized composition, CCG can generate $\{a^n a'^m b^n c^n b'^m c'^m d'^m d^n\}$, which cannot be generated by a TAG or LIG, and Hoffman (Hoffman, 1993) shows that a CCG with $\mathbf{B}_\times{}^2$ and no bounds on the arity of type-raised categories can derive $a^n b^n c^n d^n e^n$, which also cannot be generated by a TAG or LIG.

In English, type-raising and composition allow derivations of *wh*-extraction, right node raising and argument cluster coordination in which the verbs involved have the same lexical categories as in standard sentences that do not involve non-local dependencies. In TAG, these constructions either require either additional elementary trees, or non-standard coordination rules (Sarkar and Joshi, 1996) that were not taken into account in the original equivalence proof. On the other hand, the Dutch cross-serial dependencies (without extraction or coordination[1]) and the weakly equivalent $a^n b^n$, can easily by a CCG with bounded generalized composition and without type-raising. In fact, Vijay-Shanker and Weir (1994) show that a TAG can be translated into a CCG that uses only function application and composition, but does not require type-raising.

Deriving $\mathbf{a^n b^n c^n d^n}$ The language $a^n b^n c^n d^n$ can be generated by a TAG with one auxiliary tree β_1 with yield a_bc_d (where the _'s indicate where β_1 can be adjoined again), resulting in strings of the form $a^{1..n}\_b^{n..1} c^{1..n}\_d^{n...1}$ (Joshi and Schabes, 1997). Weir (1988) gives a CCG for this language, but since his grammar assumes that the string contains n empty strings ϵ with lexical categories, we give in Figure 2 a different analysis. This grammar assigns the lexical category $(\mathsf{S\backslash A})/\mathsf{D})\backslash_4\mathsf{S})/\mathsf{C}$ to any but the leftmost b, where we have used $\backslash_4$ to indicate a modality which requires backward crossed 4-ary composition.[2]

3 CCG for a fragment of German

We follow Steedman (2000) and Hockenmaier (2006) in most of our basic analyses. German has three different word orders that depend on the clause type. Main clauses (3) are verb-second. Imperatives and questions are verb-initial (4). If a modifier or one of the objects is moved to the front, the word order becomes verb-initial (4). Subordinate and relative clauses are verb-final (5):

(3) a. *Peter gibt ihm ein Buch.*
Peter gives him a book.
b. *Ein Buch gibt Peter ihm.*
c. *dann gibt Peter ihm ein Buch.*

(4) a. *Gibt Peter ihm ein Buch?*
b. *Gib ihm ein Buch!*

(5) a. *dass Peter ihm das Buch gibt.*
b. *das Buch, das Peter ihm gibt.*

We assume that the underlying word order in main clauses is always verb-initial, and that the sententce-initial subject is in fact topicalized. We use the features S_{v1} and S_{vfin} to distinguish verbs in main and subordinate clauses. Main clauses have the feature S_{dcl}, requiring either a sentential modifier with category $\mathsf{S}_{dcl}/\mathsf{S}_{v1}$, a topicalized subject ($\mathsf{S}_{dcl}/(\mathsf{S}_{v1}/\mathsf{NP}_n)$), or a type-raised argument ($\mathsf{S}_{dcl}/(\mathsf{S}_{v1}\backslash\mathsf{X})$), where X can be any argument category, such as a noun phrase, prepositional phrase, or a non-finite VP.

[1] Steedman (2000, p.212) points out that generalized coordination would be required for the coordination of unboundedly long noun or verb clusters, which would require the full generative capacity of Indexed Grammars.

[2] In multimodal versions of CCG (Baldridge, 2002), modalities that are this specific are not typically assumed, but here this is required in order to avoid overgeneration. Weir (1988) gives similar constraints in his grammar.

(1) **Case 1: Two verbs with two NP arguments each**
   a. dass der Detektiv dem Klienten **den Verdächtigen des Verbrechens zu überführen** versprochen hat.
   b. dass **des Verbrechens** der Detektiv **den Verdächtigen** dem Klienten **zu überführen** versprochen hat.

(2) **Case 2: N verbs with one NP argument each**
   a. **Dieses Buch$_1$** hat$_2$ **den Kindern$_3$** *niemand$_4$* **zu geben$_5$** *versucht$_6$*.
   this book has to-the-children nobody to give tried.
   Nobody has tried to give this book to the children.
   b. dass *der Rat* *dem Pfarrer* die Menschen **der Opfer** **gedenken** zu lassen *versprochen* hat.
   that the council the priest the people the victims commemorate let promised has.
   that the council has promised the priest to let the citizens commemorate the victims.
   c. dass die Menschen **der Opfer** *dem Pfarrer der Rat* **gedenken** zu lassen *versprochen* hat.
   that the people the victims the priest the council commemorate let promised has.
   that the council has promised the priest to let the citizens commemorate the victims.

Figure 2: Non-local scrambling examples (from Rambow (1994) and Becker *et al.* (1991).

The treatment of subjects Unlike Hockenmaier (2006), we treat subjects as arguments of main verb, and assume auxiliaries are categories of the form S/S and $S\backslash S$ (with appropriate features to avoid overgeneration). Evidence for this analysis (which is similar to the standard analysis of subjects in TAG) comes from coordinations that would otherwise not be derivable (see Figure 7).

Local Scrambling In the so-called "Mittelfeld" all orders of arguments and adjuncts are potentially possible. In the following example, all 5! permutations are grammatical (Rambow, 1994):

(6) *dass [eine Firma] [meinem Onkel] [die Möbel] [vor drei Tagen] [ohne Voranmeldung] zugestellt hat.*
that [a company] [to my uncle] [the furniture] [three days ago] [without notice] delivered has.

Such local scrambling cases can easily be derived with generalized composition and type-raising. However, argument-cluster coordinations are possible with all subsets of arguments:

(7) Dir gibt Maria den Ball und Peter das Buch.
to-you gives Maria the ball and Peter the book.
To you, Maria gives the ball and Peter the book.
Dir gibt den Ball Maria und das Buch Peter.

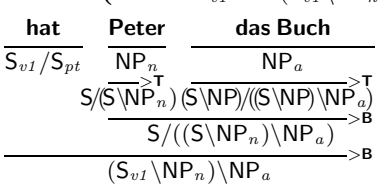
(8) Das Buch gibt Maria dir und Peter mir.
Das Buch gibt dir Maria und mir Peter.

(9) Peter gibt mir das Buch und dir den Ball.
Peter gibt das Buch mir und den Ball dir.

Like in a TAG analysis of local scrambling, we will therefore assume separate lexical categories for each possible permutation[3].

[3]To avoid this combinatorial explosion of the lexicon, extensions of CCG have been proposed (Hoffman, 1995b; Baldridge, 2002); albeit, at least in Hoffman's case, these raise its generative capacity beyond that of standard CCG

Partial VP fronting requires an analysis in which the remnant arguments in the Mittelfeld for a constituent, similar to argument cluster coordination (here $TV_{v1} = (S_{v1}\backslash NP_n)\backslash NP_a$):



Other constructions If verbs like *versprechen (promise)* have lexical categories of the form $((S\backslash NP_n)\backslash NP_d)/(S[zu]\backslash NP_n)$, with a suitable modality on the $S[zu]\backslash NP$ that requires composition, VP extraposition and the so-called Third construction can easily be derived (figure 7).

4 TAG and non-local scrambling

4.1 Non-local scrambling

Non-local scrambling, a construction in which the argument of an (arbitrarily deeply) embedded verb is moved to the matrix clause, occurs commonly in languages such as German. Becker *et al.* and Rambow (1994) show that this can result in dependencies that a standard TAG cannot capture. For instance, in sentence 2a), *das Buch* (the book), the direct object of *geben* (give), appears in the matrix clause headed by *versucht* (tried). This sentence has six segments with dependencies (1,5), (2,6), (3,5) and (4,6). It contains a discontinuous constituent 1-3-5 (*zu geben* and its objects), corresponding to an elementary tree anchored in *(zu) geben*. But in TAG, discontinuous constituents can only be created by wrapping adjunction, resulting in a string consisting of five segments (Figure 3).

Non-local Scrambling TAG adjunction

Figure 3: The dependencies in example (2a) left, and the dependencies that TAG adjunction can express (right). Blue segments are the yield of the tree that has been adjoined into the red tree.

Becker *et al.* (1991) consider two different cases which they show cannot be captured by a TAG:

1) Two verbs and four NPs TAG cannot generate $\{\sigma(NP_1{}^1, NP^2{}_1, NP^1{}_2, NP^2{}_2)V_1V_2\}$ which consists of any permutation of the two NP arguments of two verbs followed by the verbs themselves. This arises when a control verb such as *versprechen (promise)* takes a ditransitive complement such as *überführen (to prove X guilty of Y)* (see examples (1)).

2) k verbs and k NPs Becker *et al.* (1991) also consider the more general case where N verbs take one NP object each, resulting in the language $\{\sigma(NP_1, ..., NP_n)V_1...V_n\}$, and show that this not a tree-adjoining language (see examples (2)).

Based on a this observation, Becker *et al.* (1992) provide a proof that non-local scrambling of the k arguments of k verbs cannot in general be captured by Linear Context-Free Rewriting Systems, a class of formalisms to which CCG also belongs. It is, however, doubtable that an analysis of the general case is required for natural language. Joshi *et al.* (2000) show that tree-local multicomponent Tree-Adjoining Grammar (Weir, 1988), a variant of TAG that is weakly equivalent to standard TAG (and hence CCG) can deal with a limited range of scrambling cases.

5 CCG analyses for German scrambling

5.1 Non-local scrambling

1) Two verbs and four NPs The two verbs combine to a category of the form $(((\mathsf{S}\backslash\mathsf{NP}_1)\backslash\mathsf{NP}'_1)\backslash\mathsf{NP}_2)\backslash\mathsf{NP}'_2$, where the two most embedded NP_1s are arguments of the matrix verb and the two NP'_2s are arguments of the embedded verb. With $\mathbf{B}_\times{}^2$ and two lexical categories for the matrix verb (or $\mathbf{B}_\times{}^3$), all permutation orders can be derived.

2) k verbs and k NPs Under an analysis where the verb cluster forms one constituent, a category $(...(\mathsf{S}\backslash\mathsf{NP}_1)\backslash....)\backslash\mathsf{NP}_k$ is obtained. We will consider this general case in more detail below, but as can be seen from Figure 4, which gives a derivation for example 2a that cannot be derived with a standard TAG, CCG can derive more cases than TAG.

6 The equivalence of TAG and CCG revisited

Vijay-Shanker and Weir (1994) show the (weak) equivalence of TAG and CCG via a translation to head grammars and linear indexed grammars (LIG, (Gazdar, 1988)). In an indexed grammar, nonterminals are associated with stacks of indices. In a LIG, the stack associated with the LHS symbol X is copied to one of the RHS nonterminals Y, the top symbol can be popped off the stack of X, or a new symbol can be pushed onto the copy of the stack that is passed down to Y:[4]

copy: $X[\alpha] \rightarrow ...Y[\alpha]...$
pop: $X[\alpha c] \rightarrow ...Y[\alpha]...$
push: $X[\alpha] \rightarrow ...Y[\alpha c]...$

We show that translating both LTAG and CCG directly into strongly equivalent indexed grammars which capture all dependencies in the extended domain of locality via nonterminal stacks reveals that CCG requires a LIG with registers which is not strongly equivalent to any LIG that can be obtained from a LTAG.

6.1 TAG as a LIG

We define a function $\mathbf{f}$ which translates a TAG into a strongly equivalent LIG that captures all the dependencies represented within the elementary trees of the TAG via stack features (fig. 3). This function translates every local tree $X \rightarrow Y_1..Y_n$ of an elementary tree into one LIG production rule $\mathbf{f}(X) \rightarrow \mathbf{f}(Y_1)...\mathbf{f}(Y_n)$, and adds one push and one pop rule for each adjunction node. Substitution nodes and root nodes of initial trees labeled with nonterminal X are translated into a nonterminal $\mathbf{f}(X) = X[]$ with an empty stack. In order to avoid overgeneration, every internal (non-root)

[4] We will use $push_n$ and pop_n rules which push or pop n top symbols onto or off the stack as convenient abbreviations of corresponding n operations with appropriately unique nonterminals Y on the RHS.

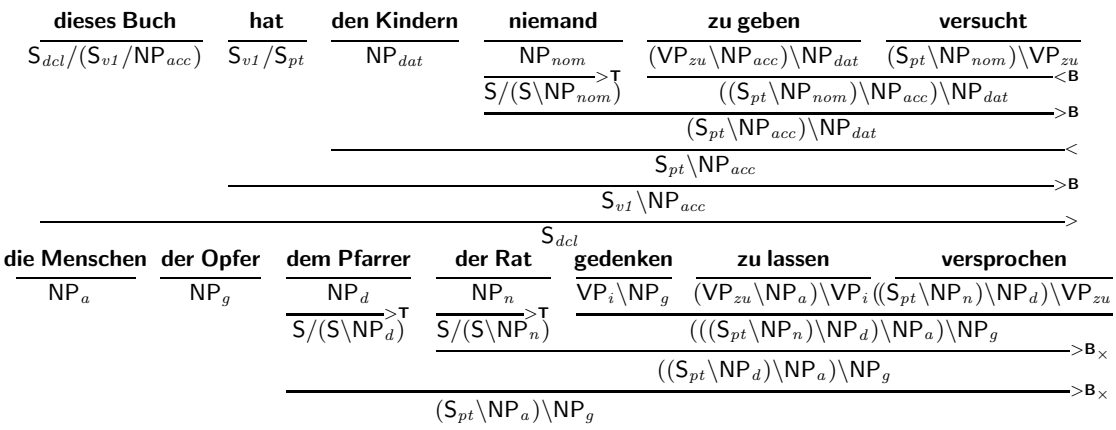


Figure 4: CCG derivations for examples (2a) and (2c) (here, VP=S\NP)

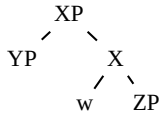
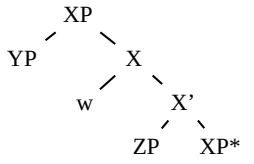
Initial trees α_j	
	$XP[] \rightarrow YP[]\ X_{\langle j,1\rangle}[\backslash yp]$ $X_{\langle j,1\rangle}[...] \rightarrow X_{\langle j,10\rangle}[.../zp]\ ZP[]$ $X_{\langle j,1\rangle}[...] \rightarrow X[...\langle j,1\rangle]$ ($X_{\langle j,1\rangle}$ is an adjunction node) $X[...\langle j,1\rangle] \rightarrow X_{\langle j,1\rangle}[...]$ ($X_{\langle j,1\rangle}$ is an adjunction node)
Auxiliary trees β_j	
	$XP[...] \rightarrow YP[]\ X_{\langle j,1\rangle}[...\backslash yp]$ $X_{\langle j,1\rangle}[...\backslash yp] \rightarrow x[\backslash yp/zp]\ X_{\langle j,11\rangle}[.../zp]$ $X_{\langle j,11\rangle}[.../zp] \rightarrow ZP[]\ X_{\langle j,111\rangle}[...]$ $X_{\langle j,1\rangle}[...] \rightarrow X[...\langle j,1\rangle]$ ($X_{\langle j,1\rangle}$ is an adjunction node) $X[...\langle j,1\rangle] \rightarrow X_{\langle j,1\rangle}[...]$ ($X_{\langle j,1\rangle}$ is an adjunction node)

Figure 5: A toy example of how TAG elementary trees are translated to LIG. The directionality of the arguments is indicated here in categorial-grammar-like notation.

node labeled X on the head path of an elementary tree is translated into a unique nonterminal $X_{\langle i,GA\rangle}$, where the index i identifies the original elementary tree and GA is the Gorn address of the corresponding node in this tree.

We assume that all nodes in an initial tree either lie on the head path from root to the single lexical anchor or are immediate descendants of a node on the head path, and that all nodes which are not on the head path are dependents of the lexical anchor. In the resulting LIG, every dependency represented by an initial tree then corresponds to a *push* operation, resulting in the lexical anchor being associated with a preterminal whose stack corresponds to its subcategorization frame:

Generating a dependent Z (initial tree):
$X[...] \rightarrow Z[]\ Y[...c]$
Generating the anchor w of an initial tree:
$X[\alpha] \rightarrow w$

Root nodes of auxiliary trees are translated into nonterminals $X[...]$ with a stack variable. This stack is passed through the productions that correspond to the auxiliary tree until it reaches the corresponding foot node, which is also translated into the same nonterminal $X[...]$ with a stack variable. The dependencies within the auxiliary tree correspond to indices which are pushed onto and popped off this stack. Every auxiliary tree defines n dependencies where the dependents have scope over the lexical anchor and m dependencies where the lexical anchor has scope over the dependents. Every dependent Y_i that has scope over the lexical anchor is generated by a rule which pushes a symbol y_i onto the stack. The anchor is generated by a rule which pops the top n elements off the stack and pushes m new elements onto the stack. These m elements are popped off the stack by the rules which generate the m dependents that have scope below the anchor. Therefore, the stack asso-

ciated with the translation of the foot node is identical to the stack of the root node. Dependencies represented by auxiliary trees correspond to push operations if the dependent has scope over the lexical anchor and to pop operations if the lexical anchor has scope over the dependent. The lexical anchor of a preterminal with arguments α_n that have scope over it and β_m arguments that have scope below it is associated with a preterminal $t[\alpha_n]$, whereas the arguments β_m that appear below it are pushed onto the stack when the lexical anchor is generated:

Generating a dependent Y with scope *over* the anchor (auxiliary tree):
$X[...] \quad \rightarrow \quad Z[] \ Y[...c]$
Generating a dependent Y with scope *below* the anchor (auxiliary tree):
$X[...c] \quad \rightarrow \quad Z[] \ Y[...]$

Every such preterminal $t[\alpha_n]$ associated with the lexical anchor of an auxiliary tree is uniquely generated by a rule (corresponding to $n+m$ LIG *pop* and *push* productions) of the following form:

Generating the preterminal for the anchor of an auxiliary tree
$X[...\alpha_n] \quad \rightarrow \quad t[\alpha_n] \ Y[...\beta_m]$

where α_n corresponds to all arguments that have been pushed onto the stack by the rules corresponding to the original auxiliary tree. Every adjunction node requires two additional unary rules which push and pop a node identifier:

push: $X_{\langle i,GA\rangle}[...] \rightarrow X[...\langle i,GA\rangle]$
pop: $X[...\langle i,GA\rangle] \rightarrow X_{\langle i,GA\rangle}[...]$

6.2 CCG as a LIG with $pop_{n,m}$

When we describe CCGs as LIGs, *Categories* c consist of a *target* t and a *stack* α: $c = t:[\alpha]$. *Stacks* $[\alpha]$ are lists of categories: $\alpha \in c^i$, with $i \geq 0$ and $|\alpha| = i$. We will write α_i for any α with length $|\alpha| = i$, and $\alpha_0 = \epsilon$. *Target categories* t are drawn from a finite set of atomic categories, $t \in \mathcal{C}_{target} = \{\mathsf{S}, \mathsf{NP}, \mathsf{PP}, ...\}$, with a designated start symbol $\mathsf{S} \in \mathcal{C}_{target}$. Ignoring order restrictions, the combinatory rules can then be written as in figure 6. Type-raised categories are not allowed to be type-raised again, and the arity of $t:[\alpha]$, $m = |\alpha|$, is typically limited to the maximal arity of lexical categories. In analyses of constructions involving non-local dependencies (including scrambling), type-raising and composition are typically applied in lockstep. If the primary category in (generalized) composition $\mathbf{B}_n$ is type-raised (with the arity of $\mathsf{T}{=}m$), the result can be viewed as a single operation $pop_{n,m}$, which allows the top $n+1$th symbol of a stack of size $m+1+n$ to be popped off the stack (Fig. 6):

Typeraising + composition :
$$\frac{\mathsf{X}/(\mathsf{X}\backslash\mathsf{Y}) \quad (\mathsf{X}\backslash\mathsf{Y})|\mathsf{Z}_1...\mathsf{Z}_n}{\mathsf{X}|\mathsf{Z}_1...\mathsf{Z}_n}\ {>}\mathbf{B}^n$$
pop$_{n,m}$:
$$\frac{\mathsf{t}:[\alpha_m\ \mathsf{t}:[\alpha_m\mathsf{c}]] \quad \mathsf{t}:[\alpha_m\mathsf{c}\beta_n]}{\mathsf{t}:[\alpha_m\beta_n]}\ {>}\mathbf{B}^n$$

The effect of $pop_{n,m}$ Bounds n on composition $\mathbf{B}_n$ and m on type-raising in CCG correspond thus to a LIG that allows all $pop_{i,j}$ operations for $i \leq n$ and $j \leq m$. Given a category $t:[c_1...c_{n+m+1}]$, a standard LIG could pop the c_is only in the reverse order $c_{n+m+1}...c_1$, whereas a LIG with $pop_{n,m}$ could also pop off c_{m+1} as the first symbol. In general, if n and m are the assumed bounds on composition and type-raising, any argument c_k with $i \leq k \leq j$ for $i = |\alpha| - n$ and $j = m+1$ can be popped of a stack α with length $|\alpha| \leq n{+}m{+}1$, generating considerably more possible strings.

7 Conclusion

By translating both CCG and LTAG into strongly equivalent Indexed Grammars, we show that CCG's strong generative capacity exceeds that of TAG in a limited way, because the CCG-IG can pop symbols off the inside of the stack when the stack size does not exceed a small finite limit. This allows CCG to handle certain scrambling cases which cannot be analyzed by a TAG. We conjecture that this effect could be captured by a linear indexed grammar with a finite number of registers where stack symbols can be stored. The LIG obtained from LTAG does not allow such operations, and is therefore somewhat less expressive. Similar to Joshi *et al.* (2000), we conjecture furthermore that the limit on the stack size, albeit small, may be close to what is needed for the cases for which reliable grammaticality judgments can be obtained.

Application	Generalized Composition $\mathbf{B}_n$	Type-raising $\mathbf{T}$	Type-raising + Composition
$\dfrac{X/Y \quad Y}{X}{>}$	$\dfrac{X/Y \quad Y\|Z_1\|...\|Z_n}{X\|Z_1\|...Z_n}{>\mathbf{B}}$	$\dfrac{X}{T/(T\backslash X)}{>\mathbf{T}}$	$\dfrac{X/(X\backslash Y) \quad (X\backslash Y)\|Z_1...Z_n}{X\|Z_1...Z_n}{>\mathbf{B}^n}$
$\dfrac{t:[\alpha u{:}[\gamma]] \quad u{:}[\gamma]}{t{:}[\alpha]}{>}$	$\dfrac{t:[\alpha u{:}[\gamma]] \quad u{:}[\gamma\beta_n]}{t{:}[\alpha\beta_n]}{>\mathbf{B}^n}$	$\dfrac{c}{t:[\alpha_m t{:}[\alpha_m c]]}{>\mathbf{T}}$	$\dfrac{t:[\alpha_m\ t:[\alpha_m c]] \quad t:[\alpha_m c\beta_n]}{t:[\alpha_m\beta_n]}{>\mathbf{B}^n}$

Figure 6: CCG's combinatory rules translated to Indexed Grammar. Greek lowercase letters α, β indicate strings of stack variables. Indices β_n and α_m indicate the length of α or β. Both typeraising and composition impose limits on β_n and α_m.

References

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

Becker, Tilman, Aravind K Joshi, and Owen Rambow. 1991. Long-distance scrambling and Tree Adjoining Grammars. In *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–26, Berlin, Germany.

Becker, Tilman, Owen Rambow, and Michael Niv. 1992. The derivational generative power of formal systems or scrambling is beyond LCFRS. Technical report, IRCS, University of Pennsylvania.

Bresnan, Joan W., Ronald M. Kaplan, Stanley Peters, and Annie Zaenen. 1982. Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13:613–636.

Gazdar, Gerald. 1988. Applicability of indexed grammars to natural languages. In Reyle, Uwe and Christian Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 69–94. Reidel, Dordrecht.

Hockenmaier, Julia. 2006. Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 505–512, Sydney, Australia, July.

Hoffman, Beryl. 1993. The formal consequence of using variables in CCG categories. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 298–300, Palo Alto, Calif. Morgan Kaufmann.

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

Hoffman, Beryl. 1995b. Integrating free word order syntax and information structure. In *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics*, pages 245–252, Dublin, March.

Joshi, Aravind K. and Yves Schabes. 1997. Tree-adjoining grammars. In Rozenberg, G. and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer, Berlin, New York.

Joshi, Aravind K., Tilman Becker, and Owen Rambow. 2000. Complexity of scrambling: A new twist to the competence-performance distinction. In Abeillé, Anne and Owen Rambow, editors, *Tree-Adjoining Grammars*, pages 167–181. CSLI.

Rambow, Owen. 1994. *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. thesis, University of Pennsylvania, Philadelphia PA.

Sarkar, Anoop and Aravind Joshi. 1996. Coordination in Tree Adjoining Grammars: Formalization and implementation. In *COLING96: Proceedings of the 16th International Conference on Computational Linguistics*, pages 610–615, Copenhagen, August.

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

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

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

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

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

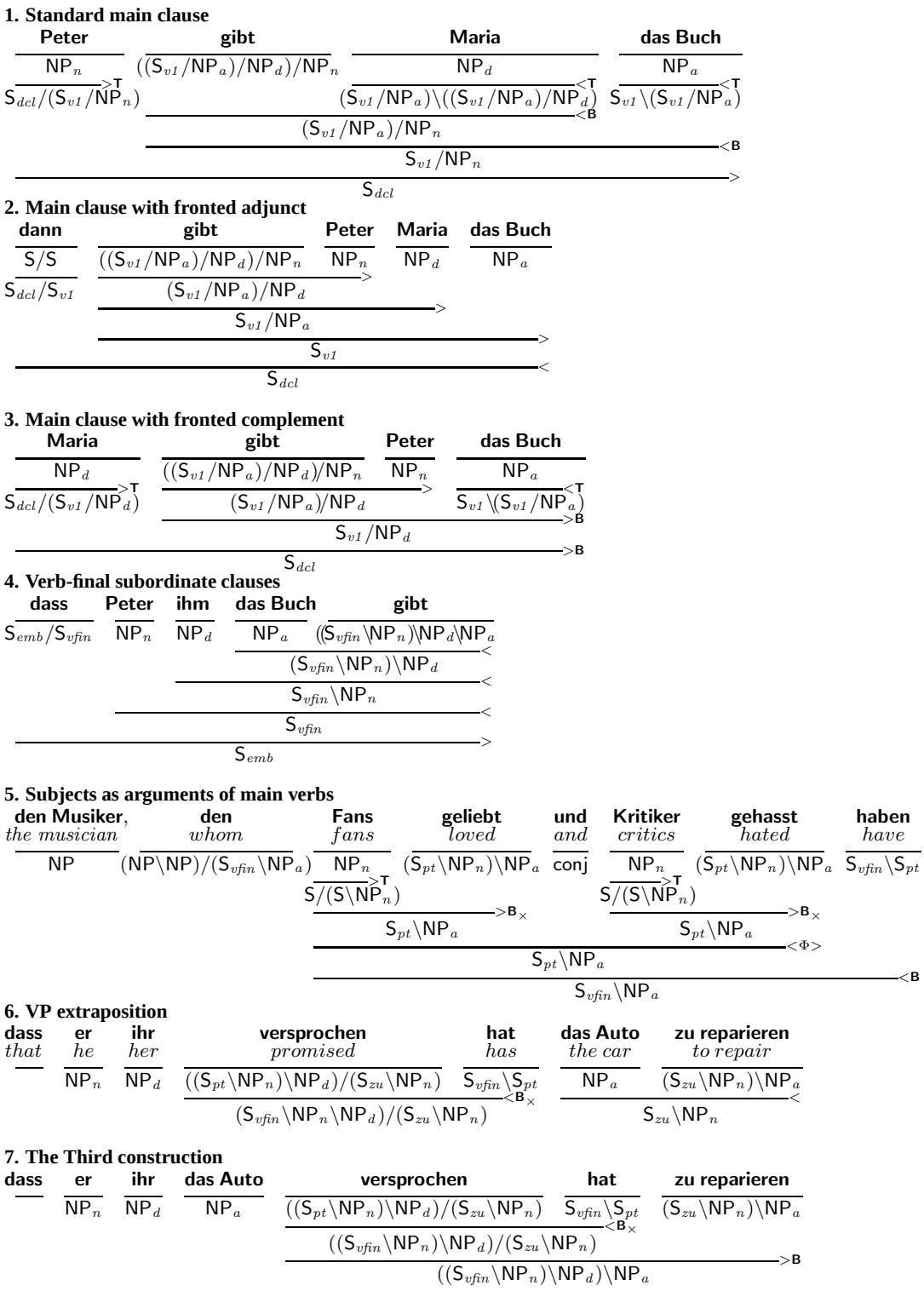


Figure 7: CCG uses topicalization (1.), a type-changing rule (2.), and type-raising (3.) to capture the different variants of German main clause order with the same lexical category for the verb, and assumes a different lexical category for verb-final subordinate clauses (4.)

Acknowledgements We would like to thank Aravind Joshi, Mark Steedman, and David Weir, as well as the anonymous TAG+ reviewers for their valuable comments and feedback.