Generative Power of CCGs with Generalized Type-Raised Categories

Nobo Komagata
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
komagata@linc.cis.upenn.edu

Abstract

This paper shows that a class of Combinatory Categorial Grammars (CCGs) augmented with a linguistically-motivated form of type raising involving variables is weakly equivalent to the standard CCGs not involving variables. The proof is based on the idea that any instance of such a grammar can be simulated by a standard CCG.

1 Introduction

The class of Combinatory Categorial Grammars (CCG-Std) was proved to be weakly equivalent to Linear Index Grammars and Tree Adjoining Grammars (Joshi, Vijay-Shanker, and Weir, 1991; Vijay-Shanker and Weir, 1994). But CCG-Std cannot handle the generalization of type raising that has been used in accounting for various linguistic phenomena including: coordination and extraction (Steedman, 1985; Dowty, 1988; Steedman, 1996), prosody (Prevost and Steedman, 1993), and quantifier scope (Park, 1995). Intuitively, all of these phenomena call for a non-traditional, more flexible notion of constituency capable of representing surface structures including "(Subj V) (Obj)" in English. Although lexical type raising involving variables can be introduced to derive such a constituent,[1] unconstrained use of variables can increase the power. For example, a grammar involving $(\mathsf{T}\backslash x)/(\mathsf{T}\backslash y)$ can generate a language $A^nB^nC^nD^nE^n$ which CCG-Std cannot (Hoffman, 1993).

This paper argues that there is a class of grammars which allows the use of linguistically-motivated form of type raising involving variables while it is still weakly equivalent to CCG-Std. A class of grammars, CCG-GTRC, is introduced in the next section as an extension to CCG-Std. Then we show that CCG-GTRC can actually be simulated by a CCG-Std, proving the equivalence.

[0]Thanks to Mark Steedman, Beryl Hoffman, Anoop Sarkar, and the reviewers. The research was supported in part by NSF Grant Nos. IRI95-04372, STC-SBR-8920230, ARPA Grant No. N66001-94-C6043, and ARO Grant No. DAAH04-94-G0426.

[1]Our lexical rules to introduce type raising are non-recursive and thus do not suffer from the problem of the overgeneration discussed in (Carpenter, 1991).

2 CCGs with Generalized Type-Raised Categories

In languages like Japanese, multiple NPs can easily form a non-traditional constituent as in "[(Subj$_1$ Obj$_1$) & (Subj$_2$ Obj$_2$)] Verb". The proposed grammars (CCG-GTRC) admit lexical type-raised categories (LTRC) of the form $\mathsf{T}/(\mathsf{T}\backslash a)$ or $\mathsf{T}\backslash(\mathsf{T}/a)$ where T is a variable over categories and a is a constant category (Const).[2] Then, composition of LTRCs can give rise to a class of categories having the form $\mathsf{T}/(\mathsf{T}\backslash a_n...\backslash a_1)$ or $\mathsf{T}\backslash(\mathsf{T}/a_n.../a_1)$, representing a multiple-NP constituent exemplified by "Subj$_1$ Obj$_1$". We call these categories **generalized type-raised categories** (GTRC) and each a_i of a GTRC an **argument** (of the GTRC).

The introduction of GTRCs affects the use of combinatory rules: functional application ">: $x/y + y \rightarrow x$" and generalized functional composition "$>B^k(_\times)$: $x/y + y|z_1...|z_k \rightarrow x|z_1...|z_k$" where k is bounded by a grammar-dependent k_{max} as in CCG-Std.[3] This paper assumes two constraints defined for the grammars and one condition stipulated to control the formal properties.

The following **order-preserving constraint**, which follows more primitive directionality features (Steedman, 1991), limits the directions of the slashes in GTRCs.

(1) In a GTRC $\mathsf{T}|_0(\mathsf{T}|_n a_n...|_1 a_1)$, the direction of $|_0$ must be the opposite to any of $|_n, ..., |_1$.

This prohibits functional composition '$>B_\times$' on 'GTRC+GTRC' pairs so that "$\mathsf{T}/(\mathsf{T}\backslash A\backslash B)$ + $\mathsf{U}\backslash(\mathsf{U}/C/D)$" does not result in $\mathsf{T}\backslash(\mathsf{T}\backslash A\backslash B/C/D)$ or $\mathsf{U}/(\mathsf{U}/C/D\backslash A\backslash B)$. That is, no *movement* of arguments across the functor is allowed.

The **variable constraint** states that:

(2) Variables are limited to the defined positions in GTRCs.

This prohibits '$>B^k(_\times)$' with $k > 1$ on the pair

[2]Categories are in the "result-leftmost" representation and associate left. Thus $a/b/c$ should be read as $(a/b)/c$ and returns a/b when an argument c is applied to its right. $A, ..., Z$ stand for nonterminals and $a, ..., z$ for complex, constant categories.

[3]There are also backward rules (<) that are analogous to forward rules (>). *Crossing* rules where z_1 is found in the direction opposite of that of y are labelled with '×'. 'k' represents the number of arguments being passed. '|' stands for a directional meta-variable for $\{/, \backslash\}$.

'Const+GTRC'. For example, '$>B^2$' on "A/B + $\mathrm{T}/(\mathrm{T}\backslash C)$" cannot realize the unification of the form "$A/B + \mathrm{T}_1|\mathrm{T}_2/(\mathrm{T}_1|\mathrm{T}_2\backslash C)$" (with $\mathrm{T} = \mathrm{T}_1|\mathrm{T}_2$) resulting in "$A|\mathrm{T}_2/(B|\mathrm{T}_2\backslash C)$".

In order to assure the expected generative capacity, we place a condition on the use of rules. The condition can be viewed in a way comparable to those on rewriting rules to define, say, context-free grammars. The **bounded argument condition** ensures that every argument category is bounded as follows:

(3) '$>B(_{\times})$' should not apply to the pair 'Const+GTRC'.

For example, this prohibits "$A/B + \mathrm{T}/(\mathrm{T}\backslash C_n...\backslash C_1) \rightarrow A/\underline{(B\backslash C_n...\backslash C_1)}$", where the underlined argument can be unboundedly large. These constraints and condition also tell us how we can implement a CCG-GTRC system without overgeneration.

The possible cases of combinatory rule application are summarized as follows:

(4) *a.* For 'Const+Const', the same rules as in CCG-Std are applicable.

b. For 'GTRC+Const', the applicable rules are:

(i) $>$: e.g., "$\mathrm{T}/(\mathrm{T}\backslash A\backslash B) + S\backslash A\backslash B \rightarrow S$"

(ii) $>B^k(_{\times})$: e.g., "$\mathrm{T}/(\mathrm{T}\backslash A\backslash B)$ + $S\backslash A\backslash B\backslash C/D \rightarrow S\backslash C/D$"

c. For 'Const+GTRC', only '$>$' is possible: e.g., "$S/(S/(S\backslash B)) + \mathrm{T}/(\mathrm{T}\backslash B) \rightarrow S$"

d. For 'GTRC+GTRC', the possibilities are:

(i) $>$: e.g., "$\mathrm{T}/(\mathrm{T}\backslash(S/A/B)) + \mathrm{T}\backslash(\mathrm{T}/A/B) \rightarrow S$"

(ii) $>B$: e.g., "$\mathrm{T}/(\mathrm{T}\backslash A\backslash B) + \mathrm{T}/(\mathrm{T}\backslash C\backslash D) \rightarrow \mathrm{T}/(\mathrm{T}\backslash A\backslash B\backslash C\backslash D)$"

CCG-GTRC is defined below where $\mathcal{G}_{std}$ and $\mathcal{G}_{gtrc}$ represent the classes of the instances of CCG-Std and CCG-GTRC, respectively:

Definition 1 $\mathcal{G}_{gtrc}$ is the collection of G's (extension of a $G \in \mathcal{G}_{std}$) such that:

1. For the lexical function f of G (from terminals to sets of categories), if $a \in f(\mathrm{a})$, f' may additionally include $\{(\mathrm{a}, \mathrm{T}/(\mathrm{T}\backslash a)), (\mathrm{a}, \mathrm{T}\backslash(\mathrm{T}/a))\}$.
2. G' may include the rule schemata in (4).

The main claim of the paper is the following:

Proposition 1 $\mathcal{G}_{gtrc}$ is weakly equivalent with $\mathcal{G}_{std}$.

We show the non-trivial direction: for any $G' \in \mathcal{G}_{gtrc}$, there is a $G'' \in \mathcal{G}_{std}$ such that $L(G') = L(G'')$. As G' corresponds to a unique $G \in \mathcal{G}_{std}$, we extend G'' from G to simulate G', then show that the languages are exactly the same.

3 Simulation of CCG-GTRC

Consider a fragment of CCG-GTRC with a lexical function f such that $f(\mathrm{a}) = \{A, \mathrm{T}/(\mathrm{T}\backslash A)\}$, $f(\mathrm{b}) = \{A, \mathrm{T}/(\mathrm{T}\backslash A)\}$, $f(\mathrm{c}) = \{S\backslash A\backslash B\}$. This fragment can generate the following two permutations:

(5) *a.*
$$\begin{array}{ccccc} \mathrm{a} & & \mathrm{b} & & \mathrm{c} \\ \mathrm{T}/(\mathrm{T}\backslash A) & + & \mathrm{T}/(\mathrm{T}\backslash B) & + & S\backslash A\backslash B \\ & & \multicolumn{3}{c}{\overline{\hspace{6em}}{>}} \\ & & \multicolumn{3}{c}{S\backslash A} \\ \multicolumn{5}{c}{\overline{\hspace{12em}}{>}} \\ \multicolumn{5}{c}{S} \end{array}$$

b.
$$\begin{array}{ccccc} \mathrm{b} & & \mathrm{a} & & \mathrm{c} \\ \mathrm{T}/(\mathrm{T}\backslash B) & + & \mathrm{T}/(\mathrm{T}\backslash A) & + & S\backslash A\backslash B \\ & & \multicolumn{3}{c}{\overline{\hspace{6em}}{>B\times}} \\ & & \multicolumn{3}{c}{S\backslash B} \\ \multicolumn{5}{c}{\overline{\hspace{12em}}{>}} \\ \multicolumn{5}{c}{S} \end{array}$$

Notice that (5*b*) cannot be generated by the original CCG-Std where the lexicon does not involve GTRCs. In order to (statically) *simulate* (5*b*) by a CCG-Std, we add $S\backslash B\backslash A$ to the value of $f''(\mathrm{c})$ in the lexicon of G''. Let us call this type of relation between the original $S\backslash A\backslash B$ and the new $S\backslash B\boxed{\backslash A}$ **wrapping**, due to its resemblance to the operation of the same name in (Bach, 1979). There are two potential problems with this simple augmentation. First, wrapping may affect unboundedly long chunks of categories as exemplified in (6). Second, the simulation may overgenerate. We discuss these issues in turn.

(6) "$\mathrm{T}/(\mathrm{T}\backslash A) + \mathrm{T}/(\mathrm{T}\backslash B) + ... + \mathrm{T}/(\mathrm{T}\backslash A) + \mathrm{T}/(\mathrm{T}\backslash B) + S\backslash A\backslash B...\backslash A\backslash B\backslash C \rightarrow S\backslash C$"

We need $S\backslash C\boxed{\backslash A\backslash B...\backslash A\backslash B}$, which can be the result of unboundedly-long compositions, to simulate (6) without depending on the GTRCs. Intuitively, this situation is analogous to long-distance movement of C from the position left of $S\backslash A\backslash B...\backslash C$ to the sentence-initial position.

In order to deal with the first problem, the following key properties of CCG-GTRC must be observed:

(7) *a.* Any derived category is a combination of lexical categories. For example, $S\backslash A\backslash B\backslash A\backslash B...\backslash A\backslash B\backslash C$ may be derived from "$S\backslash A\backslash B\backslash C$ + ... + $S\backslash A\backslash B\backslash S$ + $S\backslash A\backslash B\backslash S$" by '$<B$'.

b. Wrapping can occur only when GTRCs are involved in the use of '$>B^k_\times$' and can only *cross* at most k_{max} arguments. Since there are only finitely-many argument categories, the argument(s) being passed can be encoded in a *finite store*.

For derivable categories bounded by the maximum number of arguments of a lexical category, we add all the instances of wrapping required for simulating the effect of GTRC into the lexicon of G''. For the unbounded case, we extend the lexicon as in the following example:

(8) *a.* For a category $S\backslash A\backslash B\backslash C$, add $S_{\{\backslash C\}}\backslash A\backslash B$ to the lexicon.

b. For $S\backslash A\backslash B\backslash S$, add $S_{\{\backslash C\}}\backslash A\backslash B\backslash S_{\{\backslash C\}}$, $S\backslash A\backslash B\backslash C\backslash S_{\{\backslash C\}}, ..., S\backslash C\boxed{\backslash A\backslash B}\backslash S_{\{\backslash C\}}$.

$S_{\{\backslash C\}}$ is a new category representing the situation where $\backslash C$ is being passed across categories. Thus $\backslash C$ which originated in $S\backslash A\backslash B\backslash C$ in (*a*) may be passed onto another category in (b), after a possibly unbounded number of compositions as follows:

(9) $S_{\{\backslash C\}}\backslash A\backslash B \;+\; S_{\{\backslash C\}}\backslash A\backslash B\backslash S_{\{\backslash C\}} + \;\ldots\; +$
$S\backslash C\boxed{\backslash A\backslash B}\backslash S_{\{\backslash C\}} \rightarrow S\backslash C\boxed{\backslash A\backslash B \ldots \backslash A\backslash B\backslash A\backslash B}$

Now, both of the permutations in (5) can be derived in this extension of CCG-Std. The finite lexicon with finite extension assures the termination of the process. This covers the case (4$b$ii).

Case (4c) can be characterized by a general pattern "$c/\left(b/\left(b\backslash a_k \ldots \backslash a_1\right)\right) + \mathsf{T}/\left(\mathsf{T}\backslash a_k \ldots \backslash a_1\right) \rightarrow c$" where $\mathsf{T} = b$. Since any argument category is bounded, we can add $b/\left(b\backslash a_k \ldots \backslash a_1\right) \in f'(\mathsf{a}_1 \ldots \mathsf{a}_n)$ in the lexicon as an idiom. The other cases do not require simulation as the same string can be derived in the original grammar.

The second problem of overgeneration calls for another step. Suppose that the lexicon includes $f(\mathsf{c}) = \{S\backslash A\backslash B\}$, $f(\mathsf{d}) = \{S\backslash B\backslash A\}$, and $f(\mathsf{e}) = \left\{E\backslash\left(S\backslash B\backslash A\right)\right\}$ and that $S\backslash B\boxed{\backslash A}$ is added to $f(\mathsf{c})$ by wrapping. To avoid generating an illegal string "c e" (in addition to the legal "d e"), we label the state of wrapping as $S\backslash B_{[+wrap]}\boxed{\backslash A_{[+wrap]}}$. The original entries can be labelled as $S\backslash B_{[-wrap]}\backslash A_{[-wrap]}$ and $E\backslash\left(S\backslash B_{[-wrap]}\backslash A_{[-wrap]}\right)$. The lexical, argument categories, e.g., A, are underspecified with respect to the feature. Since finite features can be folded into a category, this can be written as a CCG-Std without features.

4 Equivalence of the Two Languages

Proposition 1 can be proved by the following lemma (as a special case where $c = S$):

Lemma 1 For any $G' \in \mathcal{G}_{gtrc}$ (an extension of G), there is a $G'' \in \mathcal{G}_{std}$ such that a string w is derivable from a constant category c in G' iff ($\leftrightarrow$) w is derivable from c in G''.

The sketch of the proof goes as follows. First, we construct G'' from G' as in the previous section. Both directions of the lemma can be proved by induction on the height of derivation. Consider the direction of '$\rightarrow$'. The base (lexical) case holds by definition of the grammars. For the induction step, we consider each case of rule application in (4). Case (4a) allows direct application of the induction hypothesis for the substructure of smaller height starting with a constant category. Other cases involve GTRC and require sublemmas which can be proved by induction on the length of the GTRC. Cases (4bi, di) have a differently-branching derivation in G'' but can be derived without simulation. Cases (4$b$ii, c) depend on the simulation of the previous section. Case (4$d$ii) only appears in sublemmas as the result category is GTRC. In each sublemma, the induction hypothesis of **Lemma 1** is applied (mutually recursively) to handle the derivations of the smaller substructures from a constant category.

A similar proof is applicable to the other direction. The special cases in this direction involves the feature $[+wrap]$ and/or the new categories of the form '$x_{\{\ldots\}}$' which record the argument(s) being passed. As before, we need sublemmas to handle each case. The proof of the sublemma involving the '$x_{\{\ldots\}}$' form can be done by induction on the length of the category.

5 Conclusion

We have shown that CCG-GTRC as formulated above is weakly equivalent to CCG-Std. The results support the use of type raising involving variables in accounting for various linguistic phenomena. Other related results to be reported in the future include: (i) an extension of the polynomial parsing algorithm of (Vijay-Shanker and Weir, 1990) for CCG-Std to CCG-GTRC (Komagata, 1997), (ii) application to a Japanese parser which is capable of handling non-traditional constituents and information structure (roughly, topic/focus structure). An extension of the formalism is also being studied, to include lexical type raising of the form $\mathsf{T}/\left(\mathsf{T}\backslash c\right)|d_1 \ldots |d_k$ for English prepositions/articles and Japanese particles.

References

Bach, Emmon. 1979. Control in Montague grammar. *Linguistic Inquiry*, 10.

Carpenter, Bob. 1991. The generative power of Categorial Grammars and Head-driven Phrase Structure Grammars with lexical rules. *Computational Linguistics*, 17.

Dowty, David. 1988. Type raising, functional composition, and non-constituent conjunction. In Richard Oehrle et al., editors, *Categorial Grammars and Natural Language Structures*. D. Reidel.

Hoffman, Beryl. 1993. The formal consequences of using variables in CCG categories. In *ACL31*.

Joshi, Aravind, K. Vijay-Shanker, and David Weir. 1991. The convergence of mildly context-sensitive grammatical formalisms. In Peter Sells et al., editors, *Foundational Issues in Natural Language Processing*. MIT Press, pages 31–81.

Komagata, Nobo. 1997. Efficient parsing of CCGs with generalized type-raised categories. Ms. University of Pennsylvania.

Park, Jong C. 1995. Quantifier scope and constituency. In *ACL33*.

Prevost, Scott and Mark Steedman. 1993. Generating contextually appropriate intonation. In *EACL6*.

Steedman, Mark J. 1985. Dependency and coordination in the grammar of Dutch and English. *Language*, 61:523–56.

Steedman, Mark. 1991. Type-raising and directionality in Combinatory Grammar. In *ACL29*.

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

Vijay-Shanker, K. and David J. Weir. 1990. Polynomial time parsing of Combinatory Categorial Grammars. In *ACL28*.

Vijay-Shanker, K. and D. J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27:511.