October 1997

Generative Power of CCGs with Generalized Type-Raised Categories

Nobo Komagata
University of Pennsylvania

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-97-15.

Generative Power of CCGs with Generalized Type-Raised Categories

Abstract
A type of 'non-traditional constituents' motivates an extended class of Combinatory Categorial Grammars (CCG), CCGs with Generalized Type-Raised Categories (CCG-GTRC) involving variables. Although the class of standard CCGs is proved to be equivalent to Linear Indexed Grammars and Tree Adjoining Grammars, use of variables can increase the power beyond these grammars. In order to establish a desired context with respect to computational complexity, this paper shows that there is a subclass of CCG-GTRC which is still weekly equivalent to the standard CCGs. The idea behind the proof is that the behavior of GTRCs can be simulated by appropriately setting the lexicon of a standard CCG.

Comments
University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-97-15.

Institute for Research in Cognitive Science

Generative Power of CCGs with Generalized Type-Raised Categories

Nobo Komagata

University of Pennsylvania
3401 Walnut Street, Suite 400A
Philadelphia, PA 19104-6228

October 1997

Site of the NSF Science and Technology Center for Research in Cognitive Science

IRCS Report 97--15

Generative Power of CCGs with Generalized Type-Raised Categories

Nobo Komagata *

Department of Computer and Information Science
and
Institute for Research in Cognitive Science
University of Pennsylvania
komagata@linc.cis.upenn.edu

September 23, 1997

Abstract

A type of 'non-traditional constituents' motivates an extended class of Combinatory Categorial Grammars (CCG), CCGs with Generalized Type-Raised Categories (CCG-GTRC) involving variables. Although the class of standard CCGs is proved to be equivalent to Linear Indexed Grammars and Tree Adjoining Grammars, use of variables can increase the power beyond these grammars. In order to establish a desired context with respect to computational complexity, this paper shows that there is a subclass of CCG-GTRC which is still weekly equivalent to the standard CCGs. The idea behind the proof is that the behavior of GTRCs can be simulated by appropriately setting the lexicon of a standard CCG.

1 Introduction

In Japanese, as in other SOV languages, a sequence of NPs can form a conjunct as exemplified below.

(1) { John-ga Mary-o } , { Ken-ga Naomi-o } tazuneta.
{ John-NOM May-ACC } CONJ { Ken-NOM Naomi-ACC } visited
"John visited Mary and Ken [visited] Naomi."

This type of 'non-traditional constituents' poses a problem to many grammar formalisms and parsers, including those specifically designed for Japanese, e.g., JPSG [Gunji, 1987] and JLE (based on finite-state syntax) [Kameyama, 1995]. Although these systems could be extended to cover the presented case, such extensions would not generalize to the wide range of non-traditional constituency.

Combinatory Categorial Grammar (CCG) has been proposed to account for non-traditional constituency in various areas of syntax [Ades and Steedman, 1982, Dowty, 1988, Steedman, 1985, Steedman, 1996] and also in the related areas of prosody, information structure, and quantifier scope [Steedman, 1991a, Prevost and Steedman, 1993, Prevost, 1995, Hoffman, 1995, Park, 1995, Park, 1996]. The mechanisms independently motivated to cover the wide range of non-traditional constituency can also provide an analysis for the NP-NP sequences in (1) as follows:

*I am grateful to Mark Steedman for his numerous suggestions and comments. I would also like to thank Beryl Hoffman, Anoop Sarkar, K. Vijay-Shanker, B. Srinivas, David Weir and reviewers of ACL/EACL-97 for their comments. 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. This technical report (IRCS-97-15) is a long version of [Komagata, 1997c] presented at ACL/EACL-97 (Student Session).

(2)
$$\dfrac{\dfrac{\text{John-ga}}{NP}\Downarrow \quad \dfrac{\text{Mary-o}}{NP}\Downarrow}{\dfrac{S/\underline{(S\backslash NP)} \quad \underline{(S\backslash NP)}/((S\backslash NP)\backslash NP)}{S/((S\backslash NP)\backslash NP)}}$$

type raising

functional composition (underlined portions are cancelled)

Informally, the NPs are assigned *higher-order function* categories associated with the basic category *NP*, and these functions can compose to derive another function category representing the NP-NP sequence.[1] The two instances of such a category can then be coordinated and take the transitive verb category, $(S\backslash NP)\backslash NP$, as the argument to derive the category *S*.

A similar type of constituents can also be formed of NPs extracted from different levels of embedding as in the following example:

(3) Japanese: Rinyouzai-wa natoriumu-ni, β syadanzai-wa koukan sinkei kei-ni,
kankei-no aru kouketuatu-no hito-ni kikimasu.
Gloss: {Diuretic-TOP sodium-DAT} & {β blocker-TOP sympathetic nervous system-DAT}
relevance-GEN exist hypertension-GEN person-DAT effective.
Translation: "Diuretic is effective for the person with hypertension related to sodium, and β blocker [is for the person with hypertension related] to sympathetic nervous system."

The underlined part is another instance of non-traditional constituent, which includes an extraction from the relative clause. Its structure is schematically shown as follows:

(4) [t_1 hypertension$_2$-GEN person-DAT effective.]
[t_2 t_3 relevance-GEN exist]

Due to functional composition, the category $(S\backslash NP)\backslash NP$, identical to the transitive verb category, can be derived for the above phrase with two extraction sites t_1 and t_3.[2] As in the first example, the NP-NP sequences at the sentence-initial position receive the category $S/((S\backslash NP)\backslash NP)$ and then the sentence category is derived after the combination with the category $(S\backslash NP)\backslash NP$.

Assuming that the competence grammar does not place a bound on the levels of embedding [Miller and Chomsky, 1963], we may have unboundedly-many extractions [Becker et al., 1991, Rambow and Joshi, 1994, Rambow, 1994]. Since no systematic constraint has been identified for the bound on the composition of such extracted constituents, we also assume that these constituents can compose without a limit, potentially resulting in an unboundedly-long NP sequence.[3] As in the case of embedding, the degraded acceptability of long sequences can be attributed to performance issues. These assumptions calls for an infinite set of type-raised categories such as $(S\backslash X_n \ldots \backslash X_1)/((S\backslash X_n \ldots \backslash X_1)\backslash NP)$ associated with NP. We capture this polymorphic situation by using variables as in $\mathsf{T}/(\mathsf{T}\backslash NP)$.[4] Although CCGs have also been motivated by psychological aspects, we will be concerned only with syntactic aspects in this paper.

The formal properties of the standard CCGs not involving variable (CCG-Std) are relatively well-studied. CCG-Std is proved weakly-equivalent to Head Grammars (HG), Linear Index Grammars (LIG), and Tree Adjoining Grammars (TAG), collectively called "mildly context-sensitive grammars" [Joshi et al., 1991, Vijay-Shanker and Weir, 1994]. CCG-Std is also shown to be polynomially parsable [Vijay-Shanker and Weir, 1990, Vijay-Shanker and Weir, 1991, Vijay-Shanker and Weir, 1993]. But, use of variables can destroy these properties. For example, Hoffman [1993]

[1] The category $S/(S\backslash NP)$ is associated with a higher-order semantics $\lambda P.Pa$ where a is the semantics of NP. Composition of $\lambda P.Pa$ and $\lambda Q.Qb$ (left to right) is $\lambda Q.Qba$. In a sense, this semantics anticipates that of a two-argument predicate (simple or complex), Q.

[2] The use of trace t is for illustration purposes only. The current approach does not assume the notion of gap or movement as the theories which employ trace.

[3] These assumptions need to be examined more carefully and is left for future research [cf. Joshi et al., 1994]. Since this paper assumes that the scope of coordination and adverb modificatioin is bounded, the unbounded composition of type-raised category actually calls for a motivation involving information structure. For example, consider the following pseudo-Japanese:

(5) Q: A-TOP B-NOM C-DAT D-NOM E-FROM who-NOM who-ACC visited-COMP heard-COMP told-COMP thought-Q?
A: $\underbrace{\text{A-TOP B-NOM C-DAT D-NOM E-FROM}}_{theme}\ \underbrace{\{\text{F-NOM G-ACC}\}\ \&\ \{\text{H-NOM I-ACC}\}\ \text{visited-COMP heard-COMP told-COMP thought.}}_{rheme}$

The 'theme' part can grow very long as can be seen. The information-structural status of the verb complex is not clear but should not affect the major division between the theme and rheme across where all the categories must be combined prior to the combination of the theme and the rheme.

[4] It is also possible to dynamically type-raise categories as needed. We take lexical approach to avoid the procedural aspects associated with dynamic type raising. Polymorphic forms also have advantage of underspecification which avoids unnecessary proliferation of categories.

showed that a grammar involving categories of the form $(\mathsf{T}\backslash x)/(\mathsf{T}\backslash y)$ can generate a language $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$ which no mildly context-sensitive grammar can generate. The use of variables in the coordination schema "x^+ **conj** $x \to x$" is also believed to generate a language $(\mathtt{wc})^n$ beyond LIG's power [Weir, 1988]. At a level higher in the scale, Becker et al. [1991], Rambow and Joshi [1994], Hoffman [1995] propose formalisms which are more powerful than mildly context-sensitive grammars to account for 'doubly'-unbounded scrambling.[5] As we know that full context-sensitive capacity is too powerful to be a formal model of natural language syntax [e.g., Savitch, 1987], it is essential to identify the generative power of the formalism which interests us.

We define a class of grammars involving variables called CCG-GTRC, and show that there is a subclass of CCG-GTRC which is weakly-equivalent to CCG-Std. Our intuition is that a linguistically-motivated use of variables in type raising can be well-constrained so that it does not increase the generative power. The key idea is that unbounded, but restricted permutations in CCG-GTRC can be 'simulated' by re-organization of categories in the lexicon. A preliminary version of the current paper appeared as [Komagata, 1997c]. The related question about parsing efficiency is addressed in [Komagata, 1997a, Komagata, 1997b].

The paper is organized as follows: Section 2 introduces Generalized Type-Raised Categories (GTRC) and defines the classes of CCG-GTRC. Section 3 proves the weak equivalence between CCG-Std and a subclass of CCG-GTRC. Section 4 includes a few simple examples.

2 CCGs with Generalized Type-Raised Categories

2.1 Generalized Type-Raised Categories

CCG-GTRC involves the class of *constant categories* (Const) and the class of *Generalized Type-Raised Categories* (GTRC).

A constant (derivable) category c can always be represented as $F|a_n\ldots|a_1$ where F is an atomic *target* category and a_i's with their directionality are *arguments*.[6] We will use "$A,\ldots,Z$" for atomic, constant categories, "$a,\ldots,z$" for possibly complex, constant categories, and '$|$' as a meta-variable for directional slashes $\{/,\backslash\}$. Categories are in the "result-leftmost" representation and associate left. Thus we usually write $F|a_n\ldots|a_1$ for $(\ldots(F|a_n)\ldots|a_1)$. We will call "$|a_i\ldots|a_j$" a *sequence* (of arguments). The length of a sequence is defined as $\left||a_i\ldots|a_1\right| = i$ while the nil sequence is defined to have the length 0. Thus an atomic constant category is considered as a category with the target category with the nil sequence. We may also use the term 'sequence' to represent an ordered set of categories such as "$c_1,\ldots,c_2$" but these two uses can be distinguished by the context. The standard CCGs (CCG-Std) solely utilize the class of Const.

GTRC is a generalization of *Lexical Type-Raised Category* (LTRC) which has the form $\overset{T}{\mathsf{T}}\left\langle\left(\mathsf{T}\right\rangle a\right)|b_i\ldots|b_1$ associated with a lexical category $a|b_i\ldots|b_1$ where $\overset{T}{\mathsf{T}}$ is a variable over categories with the atomic target category T [cf. Dowty, 1988, Steedman, 1996]. The target indication may be dropped when it is not crucial or all the atomic categories are allowed for the target. We assume the order-preserving form of LTRC using the following notation. '$\left\langle\right/$' and '$\left\backslash\right\rangle$' indicate that either set of slashes in the upper or the lower tier can be chosen but a mixture such as '$/$' and '$\backslash$' is prohibited.[7] GTRC is defined as having the form of $\mathsf{T}\left\langle(\mathsf{T}\underbrace{|a_m\ldots|a_2}_{\textit{inner sequence}}\left\rangle a_1\right)\underbrace{|b_n\ldots|b_1}_{\textit{outer sequence}}\right.$ resulting from compositions of LTRCs where $m \geq 1$, $n \geq 0$, and the directional constraint is carried over from the involved LTRCs.[8] When the directionality is not critical, we may simply write a GTRC as $\mathsf{T}|(\mathsf{T}|a_m\ldots|a_2|a_1)|b_n\ldots|b_1$. For $gtrc = \mathsf{T}|(\mathsf{T}|a_m\ldots|a_1)|b_n\ldots|b_1$, we define $|gtrc| = n+1$, ignoring the underspecified valency of the variable. Note that the introduction of LTRCs in the lexicon is non-recursive and thus does not suffer from the problem of the overgeneration discussed in [Carpenter, 1991].

[5] 'Doubly'-unbounded scrambling has the following characterization: (i) there is no bound on the distance of scrambling and (ii) there is no bound on the number of unbounded dependencies in one sentence.

[6] Note that the representation $F|a_n\ldots|a_1$ involves meta-variables for object-level constant categories but we will remain ambiguous about this object-meta distinction in this paper.

[7] For a related discussion, see [Steedman, 1991b].

[8] In a sense, GTRCs have two stacks. But these two sequences are not completely independent and does not behave like two stacks for a PDA, which is Turing-Machine equivalent.

These categories can be combined by combinatory rule schemata. Rules of (forward) "generalized functional composition" have the following form:[9]

(6) $\underset{\text{functor category}}{x/\underline{y}} \quad \blacktriangleright^k \quad \underset{\text{input category}}{\underline{y}|z_k\ldots|z_1} \quad \longrightarrow \quad \underset{\text{result category}}{x|z_k\ldots|z_1}$

The integer 'k' in this schema is bounded by k_{max}, specific to the grammar, as in CCG-Std.[10] Rules of functional application, "$x/y \blacktriangleright^0 y \rightarrow x$", can be considered as a special case of (6) where the sequence z_i's is nil.[11] The index k may be dropped when no confusion occurs. We say " '$x/\underline{y} \blacktriangleright \underline{y}|z_k\ldots|z_1$' *derives* $x|z_k\ldots|z_1$", and "$x|z_k\ldots|z_1$ *generates* the string of nonterminals '$x/y,y|z_k\ldots|z_1$' or the string of terminals '`ab`' " where the terminals `a` and `b` are associated with x/y and $y|z_k\ldots|z_1$, respectively. The case with backward rules involving '$\blacktriangleleft$' is analogous.

The use of variable for polymorphic type drew attention of researchers working on Lambek calculus [Moortgat, 1988, Emms, 1993]. In particular, Emms showed decidability for an extension called Polymorphic Lambek Calculus. The use of variables in the current formulation is limited to type raising. This reflects the intuition about the choice of rules based on 'combinators' [Steedman, 1988]. But otherwise, we do not assume that categories are wildly polymorphic.[12]

One way to represent this situation is to use two distinct subclasses of the type 'category' constructed as follows:

(7)

	Type construction	Example	
a.	*const* (Target, Arguments)	$F\backslash a_n\ldots\backslash a_1$	$\mapsto const\,(F, \backslash a_n\ldots\backslash a_1)$
b.	*gtrc* (Target, IDir, ISeq, OSeq)	$\overset{T}{\mathsf{T}}/(\mathsf{T}\backslash a_m\ldots\backslash a_1)\backslash b_n\ldots\backslash b_1$	$\mapsto gtrc\,(T, /, \backslash a_m\ldots\backslash a_1, \backslash b_n\ldots\backslash b_1)$

Such type construction can be defined in ML style as follows:[13]

(8)
```
datatype target A | B | C ...                    (* atomic categories *)
datatype dir left | right
datatype complex_cat = Complex of target * arg
     and arg = Arg of dir * complex_cat          (* mutually recursive *)
datatype seq = Seq of arg list
datatype cat = Const of target * seq
             | GTRC of target * dir * seq * seq
```

Then we can define the combinatory rules on instantiated categories. Theoretically, no unification of variable is required although our implementation based on the proposed formalism uses variable unification for convenience. Although dealing with more number of cases is tedious, the technique is straightforward. Although dealing with more number of cases is tedious, the technique is straightforward. This leads to a favorable result that CCG-GTRC is not only decidable but also polynomially recognizable [Komagata, 1997a, Komagata, 1997b].

Inclusion of GTRCs calls for thorough examination of each combinatory case depending on the involved category classes. All the possible combination of category classes are described below. Some cases are subdivided furthermore. Although the traditional categorial representation is used below, the complete description for the constructor format can be defined. A summary of the cases is given in Table 1.

(9) Const▶Const: $a/\underline{b} \blacktriangleright^k \underline{c}|d_k\ldots|d_1 \longrightarrow a|d_k\ldots|d_1$

(10) GTRC▶Const

a. Functor GTRC has an outer sequence:
$\mathsf{T}|(\mathsf{T}|a_m\ldots|a_1)|b_n\ldots|b_2/\underline{b_1} \blacktriangleright^k \underline{c}|d_k\ldots|d_1 \longrightarrow \mathsf{T}|(\mathsf{T}|a_m\ldots|a_1)|b_n\ldots|b_2|d_k\ldots|d_1$

Example: $\mathsf{T}\backslash(\mathsf{T}/PP)/\underline{NP} \blacktriangleright \underline{NP} \rightarrow \mathsf{T}\backslash(\mathsf{T}/PP)$

b. Functor GTRC has no outer sequence:

[9] Vijay-Shanker and Weir [1994] call the functor and input categories as primary and secondary components, respectively.

[10] Weir [1988] comments that the categorial grammars defined by Friedman and Venkatesan [1986] is more powerful than CCGs due to no bound on k.

[11] Forward functional composition and application are labeled as '$>B^k_{(\times)}$' and '$>$', respectively, in [Steedman, 1996].

[12] Recall that we assume conjunctives (if treated categorially) and adverbs are mapped to finite sets of categories.

[13] Note that there is no constant constructor for atomic categories.

$\mathsf{T}/\underline{(\mathsf{T}|a_m\ldots|a_2\backslash a_1)} \blacktriangleright^k \underbrace{c}_{\underline{c_0|c_m\ldots|c_1}} |d_k\ldots|d_1 \longrightarrow c_0|d_k\ldots|d_1$

Example: $\mathsf{T}/\underline{(\mathsf{T}\backslash NP\backslash NP)} \blacktriangleright \underline{S\backslash NP\backslash NP} \rightarrow S$

(11) Const▶GTRC

a. $k < |\text{input}|$:
$a/\underline{b} \blacktriangleright^k \underline{\mathsf{T}|(\mathsf{T}|c_m\ldots|c_1)|d_n\ldots|d_{k+1}}|d_k\ldots|d_1 \longrightarrow a|d_k\ldots|d_1$

Example: $(S/(S\backslash NP\backslash NP))\backslash(S/(S\backslash NP\backslash NP))/\underline{(S/(S\backslash NP\backslash NP))} \blacktriangleright \underline{\mathsf{T}/(\mathsf{T}\backslash NP\backslash NP)}$
$\rightarrow (S/(S\backslash NP\backslash NP))\backslash(S/(S\backslash NP\backslash NP))$

b. $k = |\text{input}|$ (and $k \geq 1$):
$a/\underline{b} \blacktriangleright^k \underline{\mathsf{T}|(\mathsf{T}|c_m\ldots|c_1)}|d_{k-1}\ldots|d_1 \longrightarrow a|(b\underbrace{|c_m\ldots|c_1)}_{\text{unbounded}}|d_{k-1}\ldots|d_1$

Example: $S/\underline{S} \blacktriangleright \underline{\mathsf{T}}/(\mathsf{T}\backslash NP\backslash NP) \rightarrow S/(S\backslash NP\backslash NP)$

c. $k > |\text{input}|$ (and $k \geq 2$):
$a/\underline{b} \blacktriangleright^k \underline{\mathsf{T}_0|\mathsf{T}_1|(\mathsf{T}_0|\mathsf{T}_1|c_m\ldots|c_1)}|d_{k-2}\ldots|d_1 \longrightarrow a\,|\underbrace{\mathsf{T}_1}_{\text{residual}}|(b\,|\mathsf{T}_1\,\underbrace{|c_m\ldots|c_1)}_{\text{unbounded}}|d_{k-2}\ldots|d_1$[14]

(12) GTRC▶GTRC

a. Functor GTRC has an outer sequence *and* $|\text{input}| > k$:
$\mathsf{T}|(\mathsf{T}|a_m\ldots|a_1)|b_n\ldots|b_2/\underline{b_1} \blacktriangleright^k \underline{\mathsf{U}|(\mathsf{U}|c_p\ldots|c_1)|d_n\ldots|d_{k+1}}|d_k\ldots|d_1$

$\longrightarrow \mathsf{T}|(\mathsf{T}|a_m\ldots|a_1)|b_n\ldots|b_2|d_k\ldots|d_1$

b. Functor GTRC has an outer sequence *and* $|\text{input}| = k$ (and $k \geq 1$):
$\mathsf{T}|(\mathsf{T}|a_m\ldots|a_1)|b_n\ldots|b_2/\underline{b_1} \blacktriangleright^k \underline{\mathsf{U}}|(\mathsf{U}|c_p\ldots|c_1)|d_{k-1}\ldots|d_1$

$\longrightarrow \mathsf{T}|(\mathsf{T}|a_m\ldots|a_1)|b_n\ldots|b_2|(b_1\underbrace{|c_p\ldots|c_1)}_{\text{unbounded}}|d_{k-1}\ldots|d_1$

c. Functor GTRC has an outer sequence *and* $|\text{input}| < k$ (and $k \geq 2$):
$\mathsf{T}|(\mathsf{T}|a_m\ldots|a_1)|b_n\ldots|b_2/\underline{b_1} \blacktriangleright^k \underline{\mathsf{U}_0}|\mathsf{U}_1|(\mathsf{U}_0|\mathsf{U}_1|c_p\ldots|c_1)|d_{k-2}\ldots|d_1$

$\longrightarrow \mathsf{T}|(\mathsf{T}|a_m\ldots|a_1)|b_n\ldots|b_2\,|\underbrace{\mathsf{U}_1}_{\text{residual}}|(b_1\,|\mathsf{U}_1\,\underbrace{|c_p\ldots|c_1)}_{\text{unbounded}}|d_{k-2}\ldots|d_1$

d. The functor GTRC has no outer sequence *and* $|\text{input}| > k$:

(i) T spans greater than U ($\mathsf{T} = \mathsf{U}|(\mathsf{U}|c_p\ldots|c_1)|d_n\ldots|d_{k+m+1}$):[15]
$\mathsf{T}/\underline{(\mathsf{T}|a_m\ldots|a_2\backslash a_1)} \blacktriangleright^k \underbrace{\mathsf{U}|(\mathsf{U}|c_p\ldots|c_1)|d_n\ldots|d_{k+m+1}}_{\mathsf{T}}\underbrace{|d_{k+m}\ldots|d_{k+1}}_{|a_m\ldots\backslash a_1}|d_k\ldots|d_1$

$\longrightarrow \underbrace{\mathsf{U}|(\mathsf{U}|c_p\ldots|c_1)}_{\text{inner seq of GTRC}}|d_n\ldots|d_{k+m+1}|d_k\ldots|d_1$

Example: $\mathsf{T}/\underline{(\mathsf{T}\backslash NP)} \blacktriangleright \underline{\mathsf{U}/(\mathsf{U}\backslash PP)\backslash NP} \rightarrow \mathsf{U}/(\mathsf{U}\backslash PP)$

(ii) T spans no greater than U ($\mathsf{T}|a_m\ldots|a_{m-j+1} = \mathsf{U}$):
$\mathsf{T}/(\mathsf{T}|a_m\ldots|\underbrace{a_{m-j}}_{\underline{a_{m-j,0}|a_{m-j,p}\ldots|a_{m-j,1}}}\ldots|a_2\backslash a_1) \blacktriangleright^k \underbrace{\mathsf{U}_0|\underbrace{\mathsf{U}_j\ldots|\mathsf{U}_1|(\mathsf{U}_0|\mathsf{U}_j\ldots|\mathsf{U}_1|c_p\ldots|c_1)}_{|a_m\ldots\ a_{m-j}=F|a_{(m-j,q)}\ldots|a_{(m-j,1)}}}_{\mathsf{T}}\underbrace{|d_n\ldots|d_{k+1}}_{\ldots\backslash a_1}|d_k\ldots|d_1$

$\longrightarrow F|a_{(m-j,q)}\ldots|a_{(m-j,q-j-p)}|d_k\ldots|d_1 \qquad \text{where } q \geq j+p$

[14] T could also be decomposed into $\mathsf{T}_0|\mathsf{T}_k\ldots|\mathsf{T}_1$ for a larger k but all of them share the same characteristics with the above scheme.
[15] We only consider the most general unifier.

	Case	Functor cat		Input cat		Result cat		
		Class	Outer seq	Class	$\lvert\text{input}\rvert \lessgtr k$	Class	Residual variable	Unbounded const argument
	9	Const	-	Const	$\geq$	Const	no	no
	10*a*	GTRC	yes	Const	$\geq$	GTRC	no	no
	10*b*	GTRC	no	Const	$\geq$	Const	no	no
	11*a*	Const	-	GTRC	$>$	Const	no	no
	11*b*	Const	-	GTRC	$=$	Const	no	possible
$*$	11*c*	Const	-	GTRC	$<$	neither	yes	possible
	12*a*	GTRC	yes	GTRC	$>$	GTRC	no	no
	12*b*	GTRC	yes	GTRC	$=$	GTRC	no	possible
$*$	12*c*	GTRC	yes	GTRC	$<$	neither	yes	possible
	12*d*i	GTRC	no	GTRC	$>$	GTRC	no	no
	12*d*ii	GTRC	no	GTRC	$>$	Const	no	no
	12*e*	GTRC	no	GTRC	$=$	GTRC	no	no
$*$	12*f*	GTRC	no	GTRC	$<$	neither	yes	possible

Table 1: **Combinatory Cases for CCG-GTRC**

Example: $\mathsf{T}/\underline{(\mathsf{T}\backslash(S/NP))} \blacktriangleright \underline{\mathsf{U}\backslash(\mathsf{U}/NP)} \rightarrow S$

e. The functor GTRC has no outer sequence *and* $|\text{input}| = k$ (and $k \geq 1$):
$\mathsf{T}/\underline{(\mathsf{T}|a_m\ldots|a_2\backslash a_1)} \blacktriangleright^k \underline{\mathsf{U}}|(\mathsf{U}|c_p\ldots|c_1)|d_{k-1}\ldots|d_1 \longrightarrow \mathsf{T}|(\mathsf{T}\underbrace{|a_m\ldots|a_2\backslash a_1|c_p\ldots|c_1)}_{\text{inner seq of GTRC}}|d_{k-1}\ldots|d_1$

Example: $\mathsf{T}/\underline{(\mathsf{T}\backslash NP)} \blacktriangleright \underline{\mathsf{U}}/(\mathsf{U}\backslash NP) \rightarrow \mathsf{T}/(\mathsf{T}\backslash NP\backslash NP)$

f. The functor GTRC has no outer sequence *and* $|\text{input}| < k$ (and $k \geq 2$):
$\mathsf{T}/\underline{(\mathsf{T}|a_m\ldots|a_2\backslash a_1)} \blacktriangleright^k \underline{\mathsf{U}_0}|\mathsf{U}_1|(\mathsf{U}_0|\mathsf{U}_1|c_p\ldots|c_1)|d_{k-2}\ldots|d_1$

$\longrightarrow \mathsf{T}\underset{\substack{\uparrow\\ \text{residual}}}{|\mathsf{U}_1|}(\mathsf{T}|a_m\ldots|a_2\backslash a_1|\mathsf{U}_1\underbrace{|c_p\ldots|c_1)}_{\text{unbounded}}|d_{k-2}\ldots|d_1$

The three cases indicated by '$*$' in Table 1 introduce categories which are neither Const nor GTRC due to the residual variables. This is an unintended, accidental use of functional composition. The closure of the system must be maintained by excluding these cases by the following condition:

(13) **Closure Condition**: The rule "$x \blacktriangleright^k y$" must satisfy $|y| \geq k$.

Note that the distinction between constant categories and GTRCs must be made. This condition is particularly important for implementation since the residual variables can behave beyond our imagination and the parser must be able to compute the length of a category distinctively for constant categories and GTRCs.

2.2 CCG-GTRC

We define the most general form of CCG-GTRC$_0$ as follows:[16]

Definition 1 A CCG-GTRC$_0$ is a five tuple (V_N, V_T, S, f, R) where

- V_N is a finite set of nonterminals (atomic categories)
- V_T is a finite set of terminals (lexical items, written as $\mathtt{a}, \ldots, \mathtt{z}$)

[16]For the moment, we use the term 'CCG-GTRC' ambiguously between for the class of grammars and for a grammar instance. This situation will be rectified shortly by using different style, *G* and *G*, for a class and and a member where the distinction is necessary.

- S is a distinguished member of V_N
- T is a countable set of variables[17]
- f is a function that maps elements of V_T to finite subsets of "Const $\cup$ LTRC"[18]
- R is a finite set of rule instances of Generalized Functional Composition observing Closure Condition (i.e., those summarized in Table 1 except for those with '$*$').[19]

CCG-GTRC$_0$ differs from CCG-Std in some crucial respects.

(14) *a*. The set of arguments is not bounded. Not only the inner sequence of GTRC is unbounded, but also an argument of a constant category can be unboundedly long.

b. Combinatory rules cannot be specified in a 'finite' manner as described in [Vijay-Shanker and Weir, 1994].[20] The reason is that both functor and input categories can be unboundedly long unlike CCG-Std.

From both complexity and parsing points of view, this situation seems to require more 'power' to deal with. The conjecture is that this grammar is not equivalent to CCG-Std nor polynomially parsable. What I will do in the following is to find a subclass of CCG-GTRC$_0$ which still satisfies the original motivation and can be proved weakly-equivalent to CCG-Std. We discuss the following three problems in turn: (i) the bound of the arguments of constant categories, (ii) mixed directionality in GTRC inner sequence, and (iii) the behavior of GTRC outer sequences.

First, we want to apply the same techniques of CCG-Std to the "Const$\blacktriangleright$Const" case. For this purpose, the set of arguments must be bounded [Vijay-Shanker and Weir, 1994, Vijay-Shanker and Weir, 1990]. Thus we place a bound on the length of an argument.[21]

(15) **Bounded Argument Condition**: Every argument except for the inner sequence of GTRC must be bounded by the grammar.[22]

Then the rules indicated as 'unbounded argument' in Table 1 must be restricted to those satisfying the condition while the inner sequence of GTRCs can grow without limit. We now have the following property:

(16) The set of arguments of a constant category and the set of arguments of the inner and outer sequences of a GTRC are all finite. We denote the set of all these arguments as Args.

Another option to the Bounded Argument Condition is to place a bound on the length of GTRC inner sequence. But then we need to re-evaluate our assumption about the unbounded NP sequence and the system degenerates to CCG-Std since every instance of GTRC can be represented as a constant.

The new subclass of CCG-GTRC$_0$ is defined as follows:

Definition 2 CCG-GTRC$_{bound_arg}$ is a subclass of CCG-GTRC where the Bounded Argument Condition is observed.[23]

The second problem is with the mixed directionality of the GTRC. For example, consider a GTRC $\mathsf{T}/(\mathsf{T}/a_m\ldots/a_2\backslash a_1)$ derived from "$\mathsf{T}/(\mathsf{T}\backslash a_1) \blacktriangleleft \left(\mathsf{T}\backslash(\mathsf{T}/a_2) \blacktriangleleft \ldots \blacktriangleleft \mathsf{T}\backslash(\mathsf{T}/a_m)\right)$". This may proceed with the following derivation: "$\mathsf{T}/\underline{(\mathsf{T}/a_m\ldots/a_2\backslash a_1)} \blacktriangleright \underline{c/a_m\ldots/a_2\backslash a_1}|d_k\ldots|d_1$". Although the input category, $c/a_m\ldots/a_2\backslash a_1|d_k\ldots|d_1$, seeks the arguments $a_2,\ldots,a_m$ to its right, the arguments are actually found on the left of the category. In addition, although the GTRC

[17] Each instance of GTRC must be assigned a new variable when the GTRC is instantiated at a particular string position in order to avoid unintended variable binding.

[18] Our definition does not include the empty string in the domain of f as in [Vijay-Shanker and Weir, 1993] but unlike [Vijay-Shanker and Weir, 1994].

[19] Due to the introduction of GTRC, the rule instances may involve variables even at the first argument of the functor category and at the input category.

[20] This 'finiteness' corresponds to the instantiation of the input categories. The functor category of a combinatory rule still needs a meta-variable since categories can grow without limit.

[21] Alternatively, we can limit the instances of argument to those which can occur at a 'bounded' argument position.

[22] Formally, for the rule "$x \blacktriangleright^k y$" where (i) x is not a GTRC with $|x| = 1$ and (ii) $|y| = k$, y must be instantiated. This condition is weaker than the one stated in [Komagata, 1997c] in that it does not completely exclude the rule application potentially involving unbounded argument. This is preferrable because if we prohibit, say, the rule 11*b* altogether, the following adverbial modification is impossible: $S\underline{/S} \blacktriangleright \underline{\mathsf{T}}/(\mathsf{T}\backslash NP\backslash NP) \rightarrow S/(S\backslash NP\backslash NP)$.

[23] The polynomial recognition algorithm works for this general class [Komagata, 1997a, Komagata, 1997b].

$\mathsf{T}\backslash(\mathsf{T}/a_m)$ stands adjacent to $c/a_m\ldots/a_2\backslash a_1|d_k\ldots|d_1$, a_m is unboundedly-deep in the category $c/a_m\ldots/a_2\backslash a_1|d_k\ldots|d_1$. In a sense, this difficulty corresponds to the mixture of non-order preserving type raising and the unbounded version of generalized functional composition so that $\mathsf{T}/(\mathsf{T}/a_m)$ can combine with $s|d_k\ldots|d_1/a_m\ldots/a_2\backslash a_1$ (i.e., no limit on k_{max}). The current position is to stipulate the following condition:

(17) **Unidirectional GTRC Condition**: The inner sequence of a GTRC must have the uniform directionality as in: $\mathsf{T}\left\langle\backslash\left(\mathsf{T}\right\rangle a_m\ldots\left\rangle a_1\right)|b_n\ldots|b_1$.[24]

This condition is closely related to the linguistic aspect of long-distance 'movement' across the functor. Our motivation does not depends on these phenomena. For example, the gapping conjuncts of two underlined NPs in English, "John helped Mary, Bill, Rose." might involve $S/(S\backslash NP/NP)$ from "$S/(S\backslash NP)\blacktriangleright(S\backslash NP)\backslash((S\backslash NP)/NP)$" [Steedman, 1990].[25] But I believe that such a case is inherently bounded and does not require a GTRC involving variables. We define the following subclass:

Definition 3 CCG-GTRC$_{uni}$ is a subclass of CCG-GTRC$_{bound_arg}$ where the Unidirectional GTRC Condition is observed.

The third problem is related to 'quasi-island' condition exemplified as follows:

(18) *a*. CCG-GTRC: $S/A\blacktriangleleft\left(\mathsf{T}\backslash(\mathsf{T}/A)/B\blacktriangleright B\right)\rightarrow S$

b. CCG-GTRC: $B\overset{\blacktriangleright}{\blacktriangleleft}S/A\overset{\blacktriangleright}{\blacktriangleleft}\overset{S}{\mathsf{T}}\backslash(\mathsf{T}/A)\backslash B\rightarrow *$

c. CCG-Std: $S/A\blacktriangleleft\left(A/B\blacktriangleright B\right)\rightarrow S$

d. CCG-Std: $B\blacktriangleleft(S/A\blacktriangleright A\backslash B)\rightarrow S$

With respect to the interaction with input categories of constant class, GTRCs behave like an island. But we do not have a general way in CCG-Std proper to *exactly* capture the effect. Our next step is to exclude outer sequence from the GTRCs altogether.[26]

Definition 4 CCG-GTRC$_{no_outer}$ is a subclass of CCG-GTRC$_{uni}$ where no GTRC has outer sequence.

This limits the instances of GTRCs to a finite set since the inner argument of a GTRC is 'frozen'. It can only act as its own.[27] Although the expressiveness is greatly limited, it can still represent the example we started with in addition to the coverage of CCG-Std.

These conditions may appear unnatural. But note that they are applied when the grammar is constructed and do not change the way the grammar is used to recognize a string in CCG-GTRC. Thus they are legitimate way to 'define' subclasses of grammar. In the rest of this paper, we focus on CCG-GTRC$_{no_outer}$ and prove its weak equivalence to CCG-Std. The only relevant cases are now (9), (10*b*), (11a,b), and (12*d*ii,*e*) where no outer sequence of GTRC is present.

3 Equivalence of CCG-GTRC and CCG-Std

This section presents the proof of the equivalence of CCG-Std and CCG-GTRC$_{no_outer}$ (CCG-GTRC hereafter). Let G_{std} and G_{gtrc} be the classes of CCG-Std and CCG-GTRC, respectively. A grammar is represented by G_{index} where the subscript is optionally used to distinguish grammars. The proposition to prove is the following:

[24]Formally, for "GTRC$_1$ $\blacktriangleright^k$GTRC$_2$" where $|\text{GTRC}_1|=1$, $|\text{GTRC}_2|=k$, the directionality of the inner sequences must be identical. This condition was called "Order-Preserving Condition" in [Komagata, 1997c]. I also stated that it is derivable from the principles in [Steedman, 1991b] but it is not correct.

[25]It is not clear to me why is $>B_\times$ allowed in English in this case.

[26]A weaker restriction such as the following may seem possible: the directionality of the outer sequence must equal those of the inner sequence. But we do not have a general method of achieving this effect either.

[27]The case (12*d*ii) may result in decomposition of the inner argument in a restricted way. This will be treated as Bounded GTRC in a later section.

Proposition 1 G_{gtrc} is weakly equivalent to G_{std}.

Since any $G \in G_{std}$ is also $G \in G_{gtrc}$ by definition, we only need to show that for each $G_{gtrc} \in G_{gtrc}$, there is a $G_{std} \in G_{std}$ such that G_{gtrc} and G_{std} generate the same language, i.e., $L(G_{gtrc}) = L(G_{std})$. The proof is by the following lemma with the start category set to S.

Lemma 1 (Main Lemma) For any $G_{gtrc} \in G_{gtrc}$, there is a $G_{std-sim} \in G_{std}$ such that a terminal string $\mathtt{w}$ is generated by a constant category c in G_{gtrc} iff $\mathtt{w}$ is generated by c' in $G_{std-sim}$ where c' is the category in $G_{std-sim}$ corresponding to c in G_{gtrc}.

We will construct $G_{std-sim}$ from G_{gtrc} so that $G_{std-sim}$ *simulates* G_{gtrc}.[28] The process starts by translating G_{gtrc} to the base CCG-Std, $G_{std-base}$ as follows:

(19) *a*. Copy all the constant categories in G_{gtrc} to $G_{std-base}$ assigned to the same terminal symbol.

b. For each LTRC $\mathsf{T}\left\langle\left(\mathsf{T}\right\rangle a\right)$ in G_{gtrc}, add an atomic category $\langle a\rangle$ to the lexicon of $G_{std-base}$ assigned to the same terminal symbol.[29] Note that the use of atomic category is to avoid decomposition of the inner argument. This is possible since the inner arguments of GTRC never unifies with a target category and are never decomposed in the current formulation.[30]

Then $G_{std-base}$ is extended to $G_{std-sim}$ to simulate G_{gtrc}. This situation is shown schematically as follows:

(20)
$$\begin{array}{ccc} & \text{translation} & \\ G_{gtrc} & \longrightarrow & G_{std-base} \\ & \nwarrow & \downarrow \quad \text{extension} \\ & \text{simulation} & G_{std-sim} \end{array}$$

Since CCG-GTRC extends the way CCGs captures phenomena including unbounded, but restricted 'permutation', it is crucial to identify the properties of GTRCs and provide appropriate methods for simulation. Once we have the right simulation, the equivalence can be shown by the set inclusion for both directions by invoking the simulation as needed. Two simulation techniques, 'wrapping' and 'bounded GTRC', and the proof of both directions will be described in the following subsections.

3.1 Wrapping

CCG-GTRC allows permutation as observed in the following example:

(21) *a*.
$$\dfrac{\dfrac{\mathtt{a}\quad\mathtt{b}\quad\mathtt{c}}{\mathsf{T}/(\mathsf{T}\backslash A)\quad \underline{\mathsf{T}/(\underline{\mathsf{T}}\backslash B)}\quad \underline{S\backslash A}\backslash B}\,{>}}{S}\,{>}$$

b.
$$\dfrac{\dfrac{\mathtt{b}\quad\mathtt{a}\quad\mathtt{c}}{\mathsf{T}/(\mathsf{T}\backslash B)\quad \underline{\mathsf{T}/(\underline{\mathsf{T}}\backslash A)}\quad \underline{S\backslash A}\backslash B}\,{>}}{S}\,{>}$$
(permutation)

First, we attempt to simulate such a permutation by *wrapping* the arguments of a lexical category [cf. Bach, 1979, Dowty, 1979]. For example, '$\backslash A$' in $S\backslash A\backslash B$ can wrap across '$\backslash B$' with the result of $S\backslash B\backslash\langle A\rangle$. We use '$\langle\rangle$' to represent the wrapped argument as an atomic category which will unify with the GTRC-translated category also represented in the same way. This corresponds to the permutation of (21*b*) as follows:

[28] The word 'simulation' is also used to describe operations involved in the process.

[29] The directionality of LTRC can be captured by features such as '$-left$' or '$-right$'. We will ignore this aspect for simplicity.

[30] This depends on the 'no-outer sequence' condition.

(22) *a.*

b	a	c	(CCG-GTRC)
$\mathsf{T}/(\mathsf{T}\backslash B)$	$\mathsf{T}/(\underline{\mathsf{T}\backslash A})$	$\underline{S\backslash A\backslash B}$	
	$S\backslash B$		
S			

b.

b	a	c	(CCG-Std)
B	$\langle A\rangle$	$S\backslash B\backslash\langle A\rangle$	
	$S\backslash B$		
S			

The above-mentioned technique of wrapping arguments only applies to local permutation within a lexical category. But CCG-GTRC allows permutations across lexical categories as seen below.

(23)

$\mathsf{T}/(\mathsf{T}\backslash A)$	$\underline{T\backslash B}$	$S\backslash A\backslash\underline{T}$
	$S\backslash A\backslash B$	
$S\backslash B$		

Since we assume that GTRCs can compose without limit, there is no bound on the composition of the input to GTRCs.

(24)

$\mathsf{T}/\underline{(\mathsf{T}\backslash A_n\ldots\backslash A_2)}$	$\underline{T}\backslash A_1$	$\ldots$	$\underline{T}\backslash A_{n-1}\backslash T$	$S\backslash A_n\backslash\underline{T}$
			$S\backslash A_n\backslash A_{n-1}\backslash\underline{T}$	
			$\underline{S\backslash A_n\ldots\backslash A_2}\backslash A_1$	
$S\backslash A_1$				

Then, we want to obtain a wrapped category like $S\backslash A_1\backslash\langle A_n\rangle\ldots\backslash\langle A_2\rangle$. This situation can be captured by using the technique of *argument passing* as follows:[31]

(25) *a.* $S\backslash B\backslash\langle A\rangle \longleftarrow \underline{T^{\{\backslash B\}}} \blacktriangleleft S\backslash B\backslash\langle A\rangle\backslash\underline{T^{\{\backslash B\}}}$

Note: Since subscripts are frequently used for indexing the categories in this paper, the features are placed as superscript.

b. $S\backslash A_1\backslash\langle A_n\rangle\ldots\backslash\langle A_2\rangle \longleftarrow$

$$\left(T^{\{\backslash A_1\}} \blacktriangleleft \left(T^{\{\backslash A_1\}}\backslash\langle A_2\rangle\backslash T^{\{\backslash A_1\}} \blacktriangleleft \cdots \blacktriangleleft \left(\underline{T^{\{\backslash A_1\}}}\backslash\langle A_{n-1}\rangle\backslash T^{\{\backslash A_1\}} \blacktriangleleft S\backslash A_1\backslash\langle A_n\rangle\backslash\underline{T^{\{\backslash A_1\}}}\right)\right)\right)$$

The arguments which are crossed by wrapping are placed as a feature on the target category and on the first argument. They are then passed on to the category corresponding to a deeper position of the composed category. As in the case of '$\langle\rangle$', we consider the category with passed arguments as an atomic category. This also applies for the case where the canceled category is complex such as: $(S/A)^{\{/C\}}/B$.

This simulation depends on the fact that the list of passed arguments is bounded. First, observe (a) below. A particular argument can be crossed by any number of arguments by wrapping, which is the source of unbounded permutation. On the other hand, an argument can cross only finite number of arguments by wrapping as seen in (b). This latter case is bounded by k_{max} of functional composition.

(26) *a.*

B	$\mathsf{T}/(\mathsf{T}\backslash A_n)$	$\cdots$	$\mathsf{T}/(\mathsf{T}\backslash A_1)$	S	$\backslash A_n\ldots\backslash A_1$	$\vert B$	
						$\vert$	
B	$\langle A_n\rangle$	$\cdots$	$\langle A_1\rangle$	S		$\vert B$	$\backslash\langle A_n\rangle\ldots\backslash\langle A_1\rangle$

b.

$\mathsf{T}/(\mathsf{T}\backslash A)$	S	$\backslash A$	$\vert B_k\ldots\vert B_1$	
			$\vert$	
$\langle A\rangle$	S		$\vert B_k\ldots\vert B_1$	$\backslash\langle A\rangle$

Recall that the set of arguments Args is bounded. Thus at any juncture of rule application, there are only finitely many possibility of argument passing. We add all these cases to the lexicon.

[31] Argument passing is conceptually similar to the techniques found in grammar formalism and logic including: SLASH feature of GPSG/HPSG [Gazdar et al., 1985, Pollard and Sag, 1994] and assume/discharge of natural deduction [Hepple, 1990]. But it is finite and limited in its power.

To describe wrapping concisely, we introduce the following notation: Depending on how we divide a category into the 'function' and the 'arguments', a category $c = F|a_m...|a_1$ can be viewed with different valencies, i.e., $c = \underset{F}{f_m}|a_m...|a_1$,..., $c = \underset{F|a_m...|a_{i+1}}{f_i}|a_i...|a_1, c = \underset{F|a_m...|a_2}{f_1}|a_1$, $c = \underset{F|a_m...|a_1}{f_0}$. Let us refer to f_i as the *functional forms* of c. The functional forms with every valency can then be represented as follows: $c = F|a_m...|a_1 = f_i\mathbb{A}_i$ where $0 \leq i \leq m$ and $\mathbb{A}_i = |a_i...|a_1$.

The process of wrapping is now presented as follows:

(27) **Wrapping**: Consider functional forms of a lexical category $c = F|a_m...|a_1 = f_i[\mathbb{A}_i|a_1]$ where $\mathbb{A}_i = |a_i...|a_2$ and '[ ]' indicates the optionality. In case a_1 is not nil, consider all the possible sequence of arguments $\mathbb{I} = |d_k...|d_1$ (as passed arguments) where $|\mathbb{I}| \leq k_{max}$. For a concatenation of $\mathbb{A}_i\mathbb{I}$ (including $|\mathbb{I}| = 0$), apply all the possible wrapping. Note the use of $\langle a_i \rangle$ to represent the wrapped argument a_i. Optionally, designate the last $j \leq k_{max}$ arguments as $\mathbb{O}$, and place them as the feature on f_i. The process can be abbreviated as follows: $f_i\mathbb{A}_i|a_1 \Longrightarrow f_i^{\{\mathbb{O}\}}\mathbb{A}'_i|a_1^{\{\mathbb{I}\}}$ where A'_i is obtained by wrapping as described above and the both categories are assigned to the same terminal. Categories with passed arguments are considered as atomic categories.

Categories including a wrapped argument and/or a passed argument, do not interact with constant category until these features are canceled.[32] For example, the following unintended cases all fail.

(28) *a.* $\underline{D} \blacktriangleleft \underline{S\backslash B\backslash \langle D \rangle} \rightarrow *$

b. $C/\underline{(S\backslash B\backslash A)} \blacktriangleright \underline{S\backslash C\backslash \langle A \rangle} \rightarrow *$

c. $\underline{S}\backslash B \blacktriangleright \underline{S\backslash B\backslash A\backslash S^{\{\backslash B\}}} \rightarrow *$

The use of '$\langle\rangle$' avoids overgeneration of the following kind as well:

(29) *a.* $S/C/\underline{A} \blacktriangleright \underline{\left(A/B \blacktriangleright B\right)} \rightarrow S/C$ (potential overgeneration)

b. $S/C/\underline{\langle A \rangle} \blacktriangleright \underline{\left(\langle A/B \rangle \blacktriangleright B\right)} \rightarrow *$ (implemented)

3.2 Bounded GTRC

When GTRCs appear as input category, their instances are bounded (16). Thus we can replace the variables with constants. For example, suppose that coordination is lexical, defined for each instance of conjunct category, and the set of conjuncts is bounded. Coordination of non-traditional constituents might need the conjunctive category like $\left(S/\left(S\backslash NP\backslash NP\right)\right) / \left(S/\left(S\backslash NP\backslash NP\right)\right) \backslash \left(S/\left(S\backslash NP\backslash NP\right)\right)$. Then we can derive $S/\left(S\backslash NP\backslash NP\right)$ as "$S/\left(S\backslash NP\right) \blacktriangleright \left(S\backslash NP\right)/\left(S\backslash NP\backslash NP\right)$". Both of the instances must be added to the lexicon since $G_{std-sim}$ has no other way to represent this non-traditional constituency. Since we are motivated to deal with unboundedly-long inner sequence of GTRC, we cannot apply this technique to (12*e*). Wrapping has been introduced for this purpose. The procedure of adding GTRC instances is described as follows:

(30) **Bounded GTRC**:

(11*a*): Suppose that the whole GTRC $\mathsf{T}\big\langle\left(\mathsf{T}\big\rangle a_m...\big\rangle a_1\right)$ unifies with some argument of a category, i.e., a member of the set of arguments Arg in G_{gtrc}. The GTRC must be derived uniquely from a sequence of LTRCs, $\mathsf{T}_m/\left(\mathsf{T}_m\backslash a_m\right),...,\mathsf{T}_1/\left(\mathsf{T}_1\backslash a_1\right)$ or $\mathsf{T}_1\backslash\left(\mathsf{T}_1/a_1\right),...,\mathsf{T}_m\backslash\left(\mathsf{T}_m/a_m\right)$, depending on the directionality (cf. **Lemma 3**).[33] Add the ground instances of the LTRCs to the lexicon of $G_{std-sim}$.

(11*b*): Since we have set a bound on the instances on $\left(b\big\rangle c_m...\big\rangle c_1\right)$, add the LTRCs which derives $b\big\langle\left(b\big\rangle c_m...\big\rangle c_1\right)$.

(12*d*ii): The only possibility is the following: "$\mathsf{T}/\underline{(\mathsf{T}\backslash a)} \blacktriangleright \underbrace{\mathsf{U}}_{\mathsf{T}} \underbrace{|(\mathsf{U}|c_p...|c_1)}_{a=F|a_m...|a_1} \rightarrow F|a_m...|a_{m-p}|d_k...|d_1$". The functor category must be an LTRC and the instances of a is bounded. We add those instances in the lexicon.

[32] Although an additional feature was introduced to control potential overgeneration in [Komagata, 1997c], the current formulation with '$\langle\rangle$' avoids the feature and simplifies the analysis.

[33] This can be proved by induction on the length of the inner sequence.

3.3 Proof: $L(G_{gtrc}) \subseteq L(G_{std-sim})$

Now the simulation is established for the given CCG-GTRC. The proof of the direction from G_{gtrc} to $G_{std-sim}$ is by induction on the height h of a derivation in G_{gtrc}. The primary recursion (for both directions) deals only with constant categories (of CCG-GTRC) since we are concerned with derivations of a constant category, S in particular. The current direction also involves GTRCs as the source derivation and these are handled by **Lemma 3** and wrapping handled by **Lemma 4** introduced below. The latter lemma sets a mutually-recursive situation with this direction of the main lemma (**Lemma 2**).

Lemma 2 The direction $L(G_{gtrc}) \subseteq L(G_{std-sim})$ of the Main Lemma.

Base case ($h = 0$): c is a lexical category. Then c is also in $G_{std-sim}$ assigned to the same terminal symbol.

Induction hypothesis (IH2): The lemma holds for all $h' \leq h-1$.

Induction step ($h \geq 1$): We only consider the following relevant cases where the result category is Const.

(31) *a.* (Const▶Const, 9) The same derivation is available in $G_{std-sim}$. For the left and right categories, which are constant categories of smaller height, apply the induction hypothesis (IH2). The pair of strings obtained by the application of the induction hypothesis in the same order can be concatenated to provide the desired string in $G_{std-sim}$.

b. (GTRC▶Const, 10*b*) $\mathsf{T}/\underline{(\mathsf{T}\backslash a_m \ldots \backslash a_1)} \blacktriangleright \underset{\underset{c_0|c_m \ldots |c_1}{=}}{c} \quad |d_k \ldots |d_1 \longrightarrow c_0|d_k \ldots |d_1$

This case requires the simulation. Note that c is unbounded. **Lemma 4** provides us the wrapped form $c_0|d_k \ldots |d_1 \backslash a_m \ldots \backslash a_1$ from $c_0 \backslash a_m \ldots \backslash a_1 | d_k \ldots | d_1$. **Lemma 3** shows that there is a sequence of categories with the corresponding string which can combine with $c_0|d_k \ldots |d_1 \backslash a_m \ldots \backslash a_1$ in the same order with the same string. Thus, after applying each category of the sequence to the wrapped category, we have the desired result $c_0|d_k \ldots |d_1$ with the same string.

c. (Const▶GTRC, 11*a*) $a/\underline{b} \blacktriangleright \underline{\mathsf{T}|(\mathsf{T}|c_m \ldots |c_1)} \longrightarrow a$

Since the GTRC is bounded, we have the corresponding category in $G_{std-sim}$ by (30). The rest is similar to the previous case.

d. (Const▶GTRC, 11*b*) $a/\underline{b} \blacktriangleright \underline{\mathsf{T}}|(\mathsf{T}|c_m \ldots |c_1) \longrightarrow a|(b|c_m \ldots |c_1)$

Since the GTRC is bounded by the stipulated Bounded Argument Condition, we have the corresponding category in $G_{std-sim}$ by (30). The rest is similar to (a).

e. (GTRC▶GTRC, 12*d*ii) $\mathsf{T}/(\mathsf{T}\backslash \underset{\underset{a_0|a_p \ldots |a_1}{=}}{a}) \blacktriangleright \underline{\mathsf{U}|(\mathsf{U}|c_p \ldots |c_1)} \longrightarrow a_0$

Since a is bounded, the process is similar to the previous case.

■

Lemma 3 If the derivation of $\mathsf{T}/(\mathsf{T}\backslash a_m \ldots \backslash a_1)$ from the string $\mathtt{w}$ can combine with $x\backslash a_m \ldots \backslash a_1$ in G_{gtrc}, $\langle a_m \rangle, \ldots, \langle a_1 \rangle$ which is associated with the same terminal string can combines with $x\backslash y_m \ldots \backslash y_1$ for some x in $G_{std-sim}$ where y_i may be a_i or $\langle a_i \rangle$.

Proof: By induction on the height h of derivation.[34]

Base case ($h = 0$): The category must be an LTRC, $\mathsf{T}/(\mathsf{T}\backslash a)$. Thus there is $\langle a \rangle$ and a assigned to the same terminal in $G_{std-sim}$ by the simulation. Then either $\langle a \rangle$ or a can combine with $x\backslash a$ or $x\backslash \langle a \rangle$ as desired.

Induction hypothesis: The lemma holds for $h' \leq h-1$.

Induction step ($h \geq 1$): The GTRC $\mathsf{T}/(\mathsf{T}\backslash a_m \ldots \backslash a_1)$ must be derived as "$\mathsf{T}/\underline{(\mathsf{T}\backslash a_m \ldots \backslash a_{i+1})} \blacktriangleright \underline{\mathsf{U}}/(\mathsf{U}\backslash a_i \ldots \backslash a_1) \rightarrow \mathsf{T}/(\mathsf{T}\backslash a_m \ldots \backslash a_1)$" for some i (12*e*). Apply the induction hypothesis to the input category. Then we have a sequence of $\langle a_i \rangle, \ldots, \langle a_1 \rangle$ which generates the same string as $\mathsf{U}/(\mathsf{U}\backslash a_i \ldots \backslash a_1)$. Since each of $\langle a_i \rangle$ has the corresponding a_i, the

[34] Induction on the length of the inner sequence also works for this case.

sequence can apply to $x'\backslash y_i \ldots \backslash y_1$ in series to derive some $x' = x\backslash y_m \ldots \backslash y_{i+1}$ in $G_{std-sim}$. Next, apply the induction hypothesis to the functor category and $x\backslash y_m \ldots \backslash y_{i+1}$ to obtain x in $G_{std-sim}$ from the same string as desired.

■

Lemma 4 Consider a category $c|a_m \ldots |a_1|d_k \ldots |d_1$ derivable in G_{gtrc} where $k \leq k_{max}$ and $m \geq 0$. If this category combines with a GTRC $\mathsf{T}/(\mathsf{T}\backslash a_m \ldots \backslash a_1)$ to reduce to "$\mathsf{T}/(\mathsf{T}\backslash a_m \ldots \backslash a_1) \blacktriangleright c\backslash a_m \ldots \backslash a_1|d_k \ldots |d_1 \rightarrow c|d_k \ldots |d_1$", then there is a category $c|d_k \ldots |d_1\backslash y_m \ldots \backslash y_1$ in $G_{std-sim}$ where y_i is either a_i or $\langle a_i \rangle$ which generates the same terminal string as $c\backslash a_m \ldots \backslash a_1|d_k \ldots |d_1$ in G_{gtrc}.

The proof is by the following lemma which is a more general version.

Lemma 5 Consider a category $x := c\backslash a_m \ldots \backslash a_1|d_{k-1} \ldots |d_1|e$ in G_{gtrc} where $k \geq 1$, $k \leq k_{max}$, $m \geq 0$, '$|e$' may be nil. c and e may be associated with passed arguments as a feature. If "$\mathsf{T}/(\mathsf{T}\backslash a_m \ldots \backslash a_1) \blacktriangleright c\backslash a_m \ldots \backslash a_1|d_{k-1} \ldots |d_1|e \rightarrow c|d_{k-1} \ldots |d_1|e$", there is a category $y := c^{\{\mathbb{O}\}}|b_j \ldots |b_1 \backslash \langle a_m \rangle \ldots \backslash \langle a_1 \rangle |e^{\{\mathbb{I}\}}$ in $G_{std-sim}$ where $j \leq k_{max}$, $m \geq 0$, $\mathbb{O}$ and $\mathbb{I}$ are sequences of arguments shorter than k_{max} (possibly nil) such that y derives the same terminal string as x.

Proof: By induction on the height h of derivation.

Base case ($h = 0$): $c\backslash a_m \ldots \backslash a_1|d_{k-1} \ldots |d_1|e$ is a lexical category. By (27), there is $c|d_{k-1} \ldots |d_1 \backslash \langle a_m \rangle \ldots \backslash \langle a_1 \rangle |e$ in $G_{std-sim}$ which is associated with the same terminal.

Induction hypothesis: The lemma holds for $h' \leq h-1$.

Induction step ($h \geq 1$): Consider the following cases:

(32) *a*. Reduction: $c\backslash a_m \ldots \backslash a_1|d_{k-1} \ldots |d_1|e \longleftarrow c\backslash a_m \ldots \backslash a_1|d_{k-1} \ldots |d_{p+1}/f \blacktriangleright f|d_p \ldots |d_1|e$
By induction hypothesis, $c\backslash a_m \ldots \backslash a_1|d_{k-1} \ldots |d_{p+1}/f$ has the corresponding $c|d_{k-1} \ldots |d_{p+1}|d_p \ldots |d_1 \backslash \langle a_m \rangle \ldots \backslash \langle a_1 \rangle /f^{\{|d_p \ldots |d_1\}}$ which generates the same string, and $f|d_p \ldots |d_1/e$ has the corresponding $f^{\{|d_p \ldots |d_1\}}|e$ which generates the same string. This case may involve a GTRC as the input category ($p = 0$). But such a case is limited to a bounded form. We can thus consider the bounded instances as if they are constants.

b. Reduction: $c\backslash a_m \ldots \backslash a_1|d_{k-1} \ldots |d_1|e \longleftarrow c\backslash a_m \ldots \backslash a_{i+1}/f \blacktriangleright f\backslash a_i \ldots \backslash a_1|d_{k-1} \ldots |d_1|e$
By induction hypothesis, $c\backslash a_m \ldots \backslash a_{i+1}/f$ has $c|d_{k-1} \ldots |d_1 \backslash \langle a_m \rangle \ldots \backslash \langle a_{i+1} \rangle /f^{\{|d_{k-1} \ldots |d_1\}}$, and $f \backslash \langle a_i \rangle \ldots \backslash \langle a_1 \rangle |d_{k-1} \ldots |d_1/e$ has $f^{\{|d_{k-1} \ldots |d_1\}}\backslash a_i \ldots \backslash a_1|e$.

c. Reduction: $c\backslash a_m \ldots \backslash a_1|d_{k-1} \ldots |d_1|e \longleftarrow c/f \blacktriangleright f\backslash a_m \ldots \backslash a_1|d_{k-1} \ldots |d_1|e$
By induction hypothesis, $f\backslash a_m \ldots \backslash a_1|d_{k-1} \ldots |d_1/e$ has $f|d_{k-1} \ldots |d_1 \backslash \langle a_m \rangle \ldots \backslash \langle a_1 \rangle |e$. By the induction hypothesis of the main lemma (IH2) there is a constant category which generates the same string as c/f.

■

3.4 Proof: $L(G_{gtrc}) \supseteq L(G_{std-sim})$

We will use the following classification for the categories in $G_{std-sim}$.

(33) *a*. Const_2: Categories translated from Const of G_{gtrc}. Exclusive of the following.

b. GTRC: Categories translated from GTRC of G_{gtrc}. Represented as $\langle x \rangle$.

c. Wrap: Categories obtained by Wrapping. They may include wrapped argument represented as $\langle x \rangle$ and/or passed argument $\{\mathbb{P}\}$.

d. BGTRC: Categories obtained by Bounded GTRC.

Note that 'Const_2' in this classification stands in relation to 'Const' in G_{gtrc} and that all the categories in $G_{std-sim}$ are *constant*. We will drop the subscript on Const_2 where no confusion arises.

The proof is by induction on the height h of a derivation in $G_{std-sim}$. The primary recursion is on Const and we introduce **Lemma 7** to have a mutually-recursive situation on wrapped categories.

Lemma 6 The direction $L(G_{gtrc}) \supseteq L(G_{std-sim})$ of the Main Lemma.

Base case ($h = 0$): By the definition of Const_2 above, there is a corresponding constant lexical category with the same terminal string in G_{gtrc}.

Induction hypothesis (IH6): The lemma holds for $h' \leq h-1$.

Induction step ($h \geq 1$): We consider the following cases which result in Const.

(34) *a*. Const▶Const: Apply the induction hypothesis (IH6) to the functor and input categories. Then the same strings can be generated from the corresponding categories in G_{gtrc}. Since we can apply the same rule in G_{gtrc}, we generates the same string from the same category.

b. Const▶BGTRC, BGTRC▶Const, and BGTRC▶BGTRC: By the simulation, any bounded instance of GTRC in $G_{std-sim}$ has the corresponding GTRC in G_{gtrc}. Apply IH6 to the Const. Then this case has the corresponding derivation. Note that there is no formal distinction between BGTRC and Const. Thus there may be ambiguous case where a single derivation may need be considered for both cases, where only one of them may apply.

c. Const▶Wrap, Wrap▶Const, Wrap▶BGTRC, BGTRC▶Wrap: These cases do not apply. Regardless of the position of the indication of wrapping (either $\langle x \rangle$ or passed argument), either they fail to unify with the other category or would remain in the result category.

d. GTRC▶<any class>, BGTRC▶GTRC, Const▶GTRC: Not applicable. GTRC-translated category $\langle x \rangle$ can only combine with the identical argument of a wrapped category.

e. Wrap▶Wrap: The only applicable case is the following: "$a/b^{\{\mathbb{P}\}} \blacktriangleright b^{\{\mathbb{P}\}}\mathbb{C} \rightarrow a\mathbb{C}$" (other instances of wrapping are not applicable for the same reason as (3)). Apply **Lemma 7** to both categories.

f. Wrap▶GTRC: The rule application takes the form: "$a/\langle b \rangle \blacktriangleright \langle b \rangle \rightarrow a$". By the simulation, the same string can be generated by the corresponding categories in G_{gtrc}.

■

For the case where the result category is Wrap, consider the following lemma.

Lemma 7 For a wrapped category c in $G_{std-sim}$, there is a constant category c' in G_{gtrc} which generates the same terminal string.

Proof: By induction on the height h of derivation.

Base case ($h = 0$): c is a lexical category in $G_{std-sim}$. There must be a category c' in G_{gtrc} by wrapping (27).

Induction hypothesis: The lemma holds for $h' \leq h-1$.

Induction step ($h \geq 1$):

(35) *a*. Wrap▶Wrap: The derivation takes the form: "$f^{\{\mathbb{O}\}}\mathbb{A}/b^{\{\mathbb{P}\}} \blacktriangleright b^{\{\mathbb{P}\}}\mathbb{C}|d^{\{\mathbb{I}\}} \rightarrow f^{\{\mathbb{O}\}}\mathbb{A}\mathbb{C}|d^{\{\mathbb{I}\}}$". Either $\mathbb{O}$ or $\mathbb{I}$ is non-nil. Apply the induction hypothesis to both categories. We have the corresponding $f\mathbb{O}\mathbb{A}'/b$ and $b\mathbb{P}\mathbb{C}'|d$ where $\mathbb{A}'$ and $\mathbb{C}'$ are the result of removing $\mathbb{P}$ and $\mathbb{I}$ from $\mathbb{A}$ and $\mathbb{C}$, respectively. They can derive: "$f\mathbb{O}\mathbb{A}'/b \blacktriangleright b\mathbb{P}\mathbb{C}'|d \rightarrow f\mathbb{O}\mathbb{A}'\mathbb{P}\mathbb{C}'|d = f\mathbb{O}\mathbb{A}\mathbb{C}'|d$".

b. Const▶Wrap, Wrap▶Const: Apply IH6 to Const and the induction hypothesis to Wrap. The rest is similar to the above.

c. No other case can result in Wrap.

■

4 Example of Simulation

Example 1 English heavy NP-shift

"John gave the book to Mary."

"[John gave to Mary] **the book which ....**"

$$f_{gtrc} = \left\{ \begin{array}{l} (\texttt{john}, NP), \left(\texttt{john}, \mathsf{T}\left\langle \left(\mathsf{T}\right\rangle NP\right)\right), (\texttt{the book}, NP), \left(\texttt{the book}, \mathsf{T}\left\langle \left(\mathsf{T}\right\rangle NP\right)\right), \\ (\texttt{to mary}, PP), (\texttt{to mary}, \mathsf{T}\backslash(\mathsf{T}/PP)), (\texttt{gave}, S\backslash NP/PP/NP), \ldots \end{array} \right\}$$

$$f^{base}_{std} = \left\{ \begin{array}{l} \textit{replace the GTRCs with } (\texttt{john}, \langle NP\rangle), (\texttt{the book}, \langle NP\rangle), (\texttt{to mary}, \langle PP\rangle), \\ \textit{the rest is the same} \end{array} \right\}$$

$$f^{sim}_{std} = \left\{ \begin{array}{l} \textit{add the following to the above} \\ (\texttt{gave}, S\backslash NP/NP/\langle PP\rangle), (\texttt{gave}, S/NP\backslash\langle NP\rangle/\langle PP\rangle), \ldots \end{array} \right\}$$

$$\dfrac{\dfrac{\begin{array}{cccc} \text{John} & \text{gave} & \text{to Mary} & \text{the book which ....} \\ \langle NP\rangle & S/NP\backslash\langle NP\rangle/\langle PP\rangle & \langle PP\rangle & NP \end{array}}{S/NP\backslash\langle NP\rangle}\blacktriangleleft}{S/NP}\blacktriangleleft$$

Example 2 Japanese long-distance extraction

"Mary-nom John-nom Mary-acc helped-comp thought."

"**Mary-acc** [Mary-nom John-nom helped-comp thought]."

$$f_{gtrc} = \left\{ \begin{array}{l} (\texttt{john-nom}, NP_{nom}), (\texttt{mary-nom}, NP_{nom}), (\texttt{john-acc}, NP_{acc}), (\texttt{mary-acc}, NP_{acc}), \\ (\texttt{john-nom}, \mathsf{T}/(\mathsf{T}\backslash NP_{nom})), (\texttt{mary-nom}, \mathsf{T}/(\mathsf{T}\backslash NP_{nom})), \\ (\texttt{john-acc}, \mathsf{T}/(\mathsf{T}\backslash NP_{acc})), (\texttt{mary-acc}, \mathsf{T}/(\mathsf{T}\backslash NP_{acc})), \\ (\texttt{helped}, S\backslash NP_{nom}\backslash NP_{acc}), (\texttt{comp}, S'\backslash S), (\texttt{thought}, S\backslash NP_{nom}\backslash S'), \ldots \end{array} \right\}$$

$$f^{base}_{std} = \left\{ \begin{array}{l} \textit{replace the GTRCs with } (\texttt{john-nom}, \langle NP_{nom}\rangle), (\texttt{mary-nom}, \langle NP_{nom}\rangle), \\ (\texttt{john-acc}, \langle NP_{nom}\rangle), (\texttt{mary-acc}, \langle NP_{nom}\rangle), \\ \textit{the rest is the same} \end{array} \right\}$$

$$f^{sim}_{std} = \left\{ \begin{array}{l} \textit{add the following to the above} \\ (\texttt{helped}, S\backslash NP_{acc}\backslash\langle NP_{nom}\rangle), \left(\texttt{helped}, S^{\{\backslash NP_{acc}\}}\backslash\langle NP_{nom}\rangle\right), \left(\texttt{comp}, S'^{\{\backslash NP_{nom}\}}\backslash S^{\{\backslash NP_{nom}\}}\right), \\ \left(\texttt{thought}, S\backslash NP_{nom}\backslash NP_{acc}\backslash S'^{\{NP_{acc}\}}\right), \\ \left(\texttt{thought}, S\backslash NP_{acc}\backslash\langle NP_{nom}\rangle\backslash S'^{\{NP_{acc}\}}\right), \ldots \end{array} \right\}$$

$$\dfrac{\dfrac{\begin{array}{ccccc} \text{Mary-acc} & \text{Mary-nom} & \text{John-nom} & \text{helped} & \text{-comp} \quad \text{thought} \\ NP_{acc} & \langle NP_{nom}\rangle & \langle NP_{nom}\rangle & S^{\{\backslash NP_{acc}\}}\backslash\langle NP_{nom}\rangle & S'^{\{NP_{acc}\}}\backslash S^{\{NP_{acc}\}} \quad S\backslash NP_{acc}\backslash\langle NP_{nom}\rangle\backslash S'^{\{NP_{acc}\}} \end{array}}{S\backslash NP_{nom}\backslash\langle NP_{nom}\rangle\backslash S^{\{NP_{acc}\}}}\blacktriangleleft}{S\backslash NP_{acc}\backslash\langle NP_{nom}\rangle\backslash\langle NP_{nom}\rangle}\blacktriangleleft$$

5 Conclusion

Motivated by unbounded composition of type-raised categories, we have introduced CCGs with Generalized Type-Raised Categories involving variables as an extension to the standard CCGs. Through the investigation of CCG-GTRC in detail, a subclass of CCG-GTRC is shown to be equivalent to CCG-Std. This is done by way of simulating unbounded, but restricted 'permutations' of CCG-GTRC by lexical wrapping and argument-passing across categories. This contrasts with the formalisms involving 'doubly'-unbounded scrambling, which are strictly more powerful than CCG-Std. Thus CCG-GTRC can be used in place of CCG-Std to account for non-traditional constituents including the ones shown in the introduction without proliferation of type-raised categories with the same computational properties.

The most restrictive condition for the choice of the studied subclass seems to be 'no outer sequence'. This is also associated with the limitation that the instances of GTRCs are finite. Naturally, we want $\mathsf{T}\left\langle \left(\mathsf{T}\right\rangle PP\right)/NP$ for English prepositions and $\mathsf{T}/(\mathsf{T}\backslash NP)\backslash NP$ for Japanese particles and to derive categories freely. Inclusion of outer

sequence seems to increase the power since that class cannot be simulated by CCG-Std due to the fact that CCG-Std cannot simulate certain 'island'-like behavior of GTRCs. But in practice, CCG-Std calls for additional mechanism such as conditions on rule application for various linguistic reasons. These conditions cannot be in general expressed in CCG-Std proper either. We are thus at the borderline of mildly context sensitivity. To find out where exactly we are is another question we want to ask.

The subclass *with* outer sequence but limited to unidirectional GTRCs is used as the base for an implementation of a parser which is capable of analyzing non-traditional constituents in a practical way.

References

Anthony Ades and Mark J. Steedman. 1982. On the Order of Words. *Linguistics and Philosophy*, 4.

Emmon Bach. 1979. Control in Montague Grammar. *Linguistic Inquiry*.

Tilman Becker, Aravind Joshi, and Owen Rambow. 1991. Long-Distance Scrambling and Tree Adjoining Grammars. In *EACL5*.

Bob Carpenter. 1991. The Generative Power of Categorial Grammars and Head-Driven Phrase Structure Grammars with Lexical Rules. *Computational Linguistics*, 17.

David R. Dowty. 1979. Dative 'Movement' and Thomason's Extensions of Montague Grammar. In Steven Davis and Marianne Mithun, editors, *Linguistics, Philosophy, and Montague Grammar*. University of Texas Press.

David Dowty. 1988. Type Raising, Functional Composition, and Non-Constituent Conjunction. In Richard Oehrle, Emmon Bach, and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures*. D. Reidel.

Martin Emms. 1993. Parsing with polymorphism. In *EACL6*.

Joyce Friedman and Ramarathnam Venkatesan. 1986. Categorial and Non-Categorial Languages. In *ACL24*.

G. Gazdar, E. Klein, G. K. Pullum, and I. A. Sag. 1985. *Generalized Phrase Structure Grammar*. Harvard University Press.

Takao Gunji. 1987. *Japanese Phrase Structure Grammar: A Unification-Based Approach*. D. Reidel.

Mark Hepple. 1990. *The Grammar and Processing of Order and Dependency: a Categorial Approach*. PhD thesis, University of Edinburgh.

Beryl Hoffman. 1993. The Formal Consequences of Using Variables in CCG Categories. In *ACL31*.

Beryl Hoffman. 1995. *The Computational Analysis of the Syntax and Interpretation of "Free" Word Order in Turkish*. PhD thesis, University of Pennsylvania.

Aravind Joshi, 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*, pages 31–81. MIT Press.

Aravind K. Joshi, Tilman Becker, and Owen Rambow. 1994. Complexity of Scrambling: A New Twist to the Competence - Performance Distinction. In *3e Colloque International sur les grammaires d'Arbres Adjoints*.

Megumi Kameyama. 1995. The Syntax and Semantics of the Japanese Language Engine. In Reiko Mazuka and Noriko Nagai, editors, *Japanese Sentence Processing*. Lawrence Erlbaum.

Nobo Komagata. 1997a. Efficient Parsing for CCGs with Generalized Type-Raised Categories. In *IWPT97*.

Nobo Komagata. 1997b. Efficient Parsing for CCGs with Generalized Type-Raised Categories. Technical report (IRCS-97-16), University of Pennsylvania.

Nobo Komagata. 1997c. Generative Power of CCGs with Generalized Type-Raised Categories. In *ACL35/EACL8 (Student Session)*.

George Miller and Noam Chomsky. 1963. Finitary Models of Language Users. In R.R. Luce et al., editors, *Handbook of Mathematical Psychology, Vol. II*. John Wiley and Sons.

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

Jong C. Park. 1995. Quantifier Scope and Constituency. In *ACL33*.

Jong Cheol Park. 1996. *A Lexical Theory of Quantification in Ambiguous Query Interpretations*. PhD thesis, University of Pennsylvania.

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Scott Prevost and Mark Steedman. 1993. Generating Contextually Appropriate Intonation. In *EACL6*.

Scott Prevost. 1995. *A Semantics of Contrast and Information Structure for Specifying Intonation in Spoken Language Generation*. PhD thesis, University of Pennsylvania.

Owen Rambow and Aravind K. Joshi. 1994. A Processing Model for Free Word Order Languages. In Jr. C. Clifton,

L. Frazier, and K. Rayner, editors, *Perspectives on Sentence Processing*. Lawrence Erlbaum.

Owen Rambow. 1994. *Formal and Computational Aspects of Natural Language Syntax*. PhD thesis, University of Pennsylvania.

Walter J. Savitch. 1987. Context-Sensitive Grammar and Natural Language Syntax. In Walter J. Savitch et al., editors, *The Formal Complexity of Natural Language*. D. Reidel.

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

Mark J. Steedman. 1988. Combinators and Grammars. In Richard Oehrle, Emmon Bach, and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures*. D. Reidel.

Mark J. Steedman. 1990. Gapping As Constituent Coordination. *Linguistics and Philosophy*, 13:207–2.

Mark Steedman. 1991a. Structure and Intonation. *Language*, 67.

Mark Steedman. 1991b. Type-Raising and Directionality in Combinatory Grammar. In *ACL29*.

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

K. Vijay-Shanker and David J. Weir. 1990. Polynomial Time Parsing of Combinatory Categorial Grammars. In *ACL28*.

K. Vijay-Shanker and David J. Weir. 1991. Polynomial Parsing of Extentions of Context-Free Grammars. In Masaru Tomita, editor, *Current Issues in Parsing Technology*. Kluwer.

K. Vijay-Shanker and David J. Weir. 1993. Parsing Constrained Grammar Formalisms. *Computational Linguistics*, 19(4).

K. Vijay-Shanker and D. J. Weir. 1994. The Equivalence of Four Extensions of Context-Free Grammars. *Mathematical Systems Theory*, 27:511–546.

David Weir. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania.