A Memory-Based Approach to the Treatment of Serial Verb Construction in Combinatory Categorial Grammar

Prachya Boonkwan[†‡]

† School of Informatics
University of Edinburgh
10 Crichton Street
Edinburgh EH8 9AB, UK

‡ National Electronics
and Computer Technology Center
112 Phahon Yothin Rd.
Pathumthani 12120, Thailand

Email: p.boonkwan@sms.ed.ac.uk

Abstract

CCG, one of the most prominent grammar frameworks, efficiently deals with deletion under coordination in natural languages. However, when we expand our attention to more analytic languages whose degree of pro-dropping is more free, CCG's decomposition rule for dealing with gapping becomes incapable of parsing some patterns of intra-sentential ellipses in serial verb construction. Moreover, the decomposition rule might also lead us to overgeneration problem. In this paper the composition rule is replaced by the use of memory mechanism, called **CCG-MM**. Fillers can be memorized and gaps can be induced from an input sentence in functional application rules, while fillers and gaps are associated in coordination and serialization. Multimodal slashes, which allow or ban memory operations, are utilized for ease of resource management. As a result, CCG-MM is more powerful than canonical CCG, but its generative power can be bounded by partially linear indexed grammar.

1 Introduction

Combinatory Categorial Grammar (CCG, Steedman (2000)) is a prominent categorial grammar framework. Having a strong degree of lexicalism (Baldridge and Kruijff, 2003), its grammars are encoded in terms of lexicons; that is, each lexicon is assigned with syntactic categories which dictate the syntactic derivation. One of its striking features is the combinatory operations that allow coordination of incomplete constituents. CCG is *nearly* context-free yet powerful enough for natural languages as it, as well as TAG, LIG, and HG, exhibits the lowest generative power in the mildly context-sensitive grammar class (Vijay-Shanker and Weir, 1994).

CCG accounts for gapping in natural languages as a major issue. Its combinatory operations resolve deletion under coordination, such as right-node raising (SV&SVO) and gapping (SVO&SO). In case of gapping, a specialized rule called *decomposition* is used to handle with forward gapping (Steedman, 1990) by extracting the filler required by a gap from a complete constituent.

However, serial verb construction is a challenging topic in CCG when we expand our attention to more analytic languages, such as Chinese and Thai, whose degree of pro-dropping is more free.

In this paper, I explain how we can deal with serial verb construction with CCG by incorporating memory mechanism and how we can restrict the generative power of the resulted hybrid. The integrated memory mechanism is motivated by anaphoric resolution mechanism in Categorial Type Logic (Hendriks, 1995; Moortgat, 1997), Type Logical Grammar (Morrill, 1994; Jäger, 1997; Jäger, 2001; Oehrle, 2007), and CCG (Jacobson, 1999), and gap resolution in Memory-Inductive Categorial Grammar (Boonkwan and Supnithi, 2008), as it is designed for associating fillers and gaps found in an input sentence. Theoretically, I discuss how this hybrid efficiently helps us deal with serial verb construction and how far the generative power grows after incorporating the memory mechanism.

Outline: I introduce CCG in §2, and then motivate the need of memory mechanism in dealing with serial verb construction in CCG in §3. I describe the hybrid model of CCG and the filler-gap memory in §4. I then discuss the margin of generative power introduced by the memory mechanism in §5. Finally, I conclude this paper in §6.

Proceedings of the EACL 2009 Student Research Workshop, pages 10–18,
Athens, Greece, 2 April 2009.

2 Combinatory Categorial Grammar

CCG is a lexicalized grammar; i.e. a grammar is encoded in terms of lexicons assigned with one or more syntactic categories. The syntactic categories may be atomic elements or curried functions specifying linear directions in which they seek their arguments. A word is assigned with a syntactic category by the turnstile operator $\vdash$. For example, a simplified English CCG is given below.

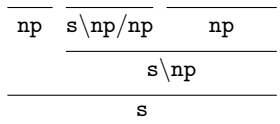
(1) John $\vdash$ np sandwiches $\vdash$ np
eats $\vdash$ s\np/np

The categories X\Y (and X/Y) denotes that X seeks the argument Y from the left (right) side.

Combinatory rules are used to combine words forming a derivation of a sentence. For basic combination, forward (>) and backward (<) functional applications, defined in (2), are used.

(2) X/Y Y ⇒ X [>]
Y X\Y ⇒ X [<]

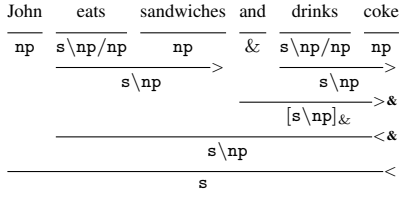
We can derive the sentence *John eats sandwiches* by the rules and the grammar in (1) as illustrated in (3). CCG is semantic-transparent; i.e. a logical form can be built compositionally in parallel with syntactic derivation. However, semantic interpretation is suppressed in this paper.

(3)


For coordination of two constituents, the coordination rules are used. There are two types of coordination rules regarding their directions: forward coordination (> **&**) and backward coordination (< **&**), defined in (4).

(4) & X ⇒ [X]$_{\&}$ [> **&**]
X [X]$_{\&}$ ⇒ X [< **&**]

By the coordination rules, we can derive the sentence *John eats sandwiches and drinks coke* in (5).

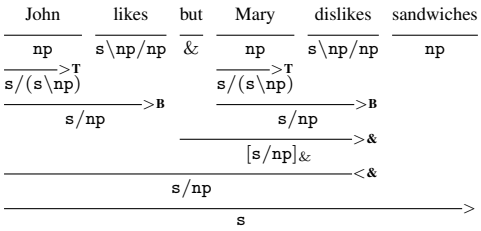
(5)


Beyond functional application and coordination, CCG also makes use of rules motivated by combinators in combinatory logics: functional composition (**B**), type raising (**T**), and substitution (**S**), namely. Classified by directions, the functional composition and type raising rules are described in (6) and (7), respectively.

(6) X/Y Y/Z ⇒ X/Z [> **B**]
Y\Z X\Y ⇒ X\Z [< **B**]

(7) X ⇒ Y/(Y\X) [> **T**]
X ⇒ Y\(Y/X) [< **T**]

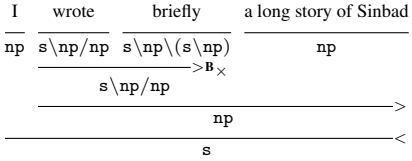
These rules permit associativity in derivation resulting in that coordination of incomplete constituents with similar types is possible. For example, we can derive the sentence *John likes but Mary dislikes sandwiches* in (8).

(8)


CCG also allows functional composition with permutation called *disharmonic functional composition* to handle constituent movement such as heavy NP shift and dative shift in English. These rules are defined in (9).

(9) X/Y Y\Z ⇒ X\Z [> **B**$_{\times}$]
Y/Z X\Y ⇒ X/Z [< **B**$_{\times}$]

By disharmonic functional composition rules, we can derive the sentence *I wrote briefly a long story of Sinbad* as (10).

(10)


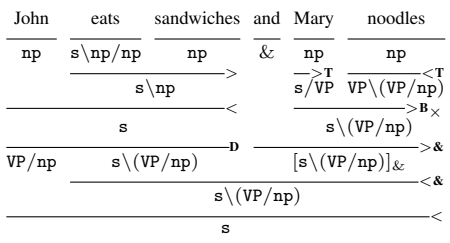
To handle the gapping coordination SVO&SO, the decomposition rule was proposed as a separate mechanism from CCG (Steedman, 1990). It decomposes a complete constituent into two parts for being coordinated with the other incomplete constituent. The decomposition rule is defined as follows.

(11) X ⇒ Y X\Y [**D**]

where Y and X\Y must be seen earlier in the derivation. The decomposition rule allows us to derive the sentence *John eats sandwiches, and Mary, noodles* as (12). Steedman (1990) stated that English is forward gapping because gapping always

takes place at the right conjunct.

(12)



where $\mathtt{VP} = \mathtt{s}\backslash\mathtt{np}$.

A multimodal version of CCG (Baldridge, 2002; Baldridge and Kruijff, 2003) restricts generative power for a particular language by annotating modalities to the slashes to allow or ban specific combinatory operations. Due to the page limitation, the multimodal CCG is not discussed here.

3 Dealing with Serial Verb Construction

CCG deals with deletion under coordination by several combinatory rules: functional composition, type raising, disharmonic functional composition, and decomposition rule. This enables CCG to handle a number of coordination patterns such as SVO&VO, SV&SVO, and SVO&SO. However, the decomposition rule cannot solve some patterns of SVC in analytic languages such as Chinese and Thai in which pro-dropping is prevalent.

The notion *serial verb construction* (SVC) in this paper means a sequence of verbs or verb phrases concatenated without connectives in a single clause which expresses simultaneous or consecutive events. Each of the verbs is marked or understood to have the same grammatical categories (such as tense, aspect, and modality), and shares at least one argument, i.e. a grammatical subject. As each verb is tensed, SVC is considered as coordination with implicit connective rather than subordination in which either infinitivization or subclause marker is made use. Motivated by Li and Thompson (1981)'s generalized form of Chinese SVC, the form of Chinese and Thai SVC is generalized in (13).

(13) $\quad (\text{Subj})\text{V}_1(\text{Obj}_1)\text{V}_2(\text{Obj}_2)\ldots\text{V}_n(\text{Obj}_n)$

The subject Subj and any objects Obj_i of the verb V_i can be dropped. If the subject or one of the objects is not dropped, it will be understood as linearly shared through the sequence. Duplication of objects in SVC is however questionable as it deteriorates the compactness of utterance.

In order to deal with SVC in CCG, I considered it syntactically similar to coordination where the connective is implicit. The serialization rule (Σ) was initially defined by imitating the forward coordination rule in (14).

(14) $\quad \mathtt{X} \;\Rightarrow\; [\mathtt{X}]_{\&} \qquad\qquad [\boldsymbol{\Sigma}]$

This rule allows us to derive by CCG some types of SVC in Chinese and Thai as exemplified in (15) and (16), respectively.

(15) wǒ zhé zhǐ zuò yí ge hézi
I fold paper make one CL box
'I fold paper to make a box.'

(16) k^{h}ǎo rîːp vîŋ k^{h}âːm t^{h}àn ǒn
he hurry run cross road
'He hurriedly runs across the road.'

One can derive the sentence (15) by considering *zhé* 'fold' and *zuò* 'make' as $\mathtt{s}\backslash\mathtt{np}/\mathtt{np}$ and applying the serialization rule in (14). In (16), the derivation can be done by assigning *rîːp* 'hurry' and *vîŋ* 'run' as $\mathtt{s}\backslash\mathtt{np}$, and *$\text{k}^{\text{h}}$âːm* 'cross' as $\mathtt{s}\backslash\mathtt{np}/\mathtt{np}$.

Since Chinese and Thai are pro-drop languages, they allow some arguments of the verbs to be pro-dropped, particularly in SVC. For example, let us consider the following Thai sentence.

(17) klâː pāj tāːm hǎː nāj râj˙ʔôi tcēː
Kla go_{DIR} $\text{follow}_{\text{V1}}$ seek_{V2} in cane-field find_{V3}
lāːj tcà dōːn tcàːk pāj
Laay FUT walk_{V4} leave_{V5} go_{DIR}
Lit: 'Kla goes out, he follows Laay (his cow), he seeks it in the cane field, and he finds that it will walk away.'
Sem: 'Kla goes out to seek Laay in the cane field and he finds that it is about to walk away.'

The sentence in (17) are split into two SVCs: the series of V_1 to V_3 and the series of V_4 to V_5, because they do not share their tenses. The directional verb *pāj* 'go' performs as an adverb identifying the outward direction of the action.

Syntactically speaking, there are two possible analyses of this sentence. First, we can consider the SVC V_4 to V_5 as a complement of the SVC V_1 to V_3. Pro-drops occur at the object positions of the verbs V_1, V_2, and V_3. On the other hand, we can also consider the SVC V_1 to V_3 and the SVC V_4 to V_5 as *adjoining construction* (Muansuwan, 2002) which indicates resultative events in Thai (Thepkanjana, 1986) as exemplified in (18).

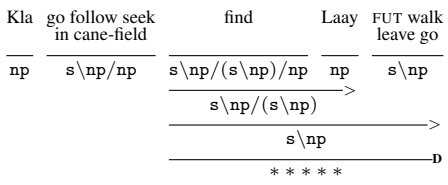
(18) pìtì tīː ŋūː tòk náːm
Piti hit snake fall water
'Piti hits a snake and it falls into the water.'

In this case, the pro-drop occurs at the subject position of the SVC V_4 to V_5, and can therefore be treated as object control (Muansuwan, 2002). However, the sentence in (17) does not show resultative events. I then assume that the first analysis is correct and will follow it throughout this paper.

We have consequently reached the question that the verb *tcɔ̄ː* 'find' should exhibit object control by taking two arguments for the object and the VP complementary, or it should take the entire sentence as an argument. To explicate the proliferation of arguments in SVC, we prefer the first choice to the second one; i.e. the verb *tcɔ̄ː* 'find' is preferably assigned as $\mathtt{s\backslash np/(s\backslash np)/np}$. In (17), the object *lāːj* 'Laay' is dropped from the verbs V_1 and V_2 but appears as one of V_3's arguments.

Let us take a closer look on the CCG analysis of (17). It is useful to focus on the SVCs of the verbs V_1-V_2 and V_3. It is shown below that the decomposition rule fails to parse the tested sentence through its application illustrated in (19).

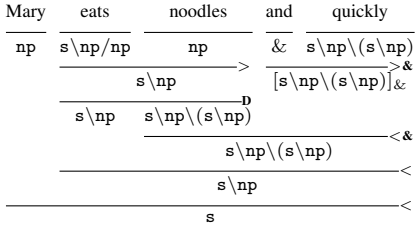
(19)



The verbs V_1 and V_2 are transitive and assigned as $\mathtt{s\backslash np/np}$, while V_4 and V_5 are intransitive and assigned as $\mathtt{s\backslash np}$. From the case (19), it follows that the decomposition rule cannot capture some patterns of intra-sentential ellipses in languages whose degree of pro-dropping is more free. Both types of intra-sentential ellipses which are prevalent in SVC of analytic languages should be captured for the sake of applicability.

The use of decomposition rule in analytic languages is not appealing for two main reasons. First, the decomposition rule does not support certain patterns of intra-sentential ellipses which are prevalent in analytic languages. As exemplified in (19), the decomposition rule fails to parse the Thai SVC whose object of the left conjunct is pro-dropped, since the right conjunct cannot be decomposed by (11). To tackle a broader coverage of intra-sentential ellipses, the grammar should rely on not only decomposition but also a supplement memory mechanism. Second, the decomposition rule allows arbitrary decomposition which leads to over-generation. From their definitions the variable $\mathtt{Y}$ can be arbitrarily substituted by any syntactic categories resulting in ungrammatical sentences generated. For example we can derive the ungrammatical sentence **Mary eats noodles and quickly* by means of the decomposition rule in (20).

(20)



The issues of handling ellipses in SVC and overgeneration of the decomposition rule can be resolved by replacing the decomposition rule with a memory mechanism that associates fillers to their gaps. The memory mechanism also makes grammar rules more manageable because it is more straightforward to identify particular syntactic categories allowed or banned from pro-dropping. I will show how the memory mechanism improves the CCG's coverage of serial verb construction in the next section.

4 CCG with Memory Mechanism (CCG-MM)

As I have elaborated in the last section, CCG needs a memory mechanism (1) to resolve intra-sentential ellipses in serial verb construction of analytic languages, and (2) to improve resource management for over-generation avoidance. To do so, such memory mechanism has to extend the generative power of the decomposition rule and improve the ease of resource management in parallel.

The memory mechanism used in this paper is motivated by a wide range of previous work from computer science to symbolic logics. The notion of memory mechanism in natural language parsing can be traced back to HOLD registers in ATN (Woods, 1970) in which fillers (antecedents) are held in registers for being filled to gaps found in the rest of the input sentence. These registers are too powerful since they enable ATN to recognize the full class of context-sensitive grammars. In Type Logical Grammar (TLG) (Morrill, 1994; Jäger, 1997; Jäger, 2001; Oehrle, 2007), Gentzen's sequent calculus was incorporated with variable quantification to resolve pro-forms and VP ellipses to their antecedents. The variable quantification in TLG is comparable to the use of memory in storing antecedents and anaphora.

In Categorial Type Logic (CTL) (Hendriks, 1995; Moortgat, 1997), gap induction was incorporated. Syntactic categories were modified with modalities which permit or prohibit gap induction in derivation. However, logical reasoning obtained from TLG and CTL are an NP-complete problem. In CCG, Jacobson (1999) attempted to explicitly denote non-local anaphoric requirement whereby she introduced the anaphoric slash (|) and the anaphoric connective (**Z**) to connect anaphors to their antecedents. However, this framework does not support anaphora whose argument is not its antecedent, such as possessive adjectives. Recently, a filler-gap memory mechanism was again introduced to Categorial Grammar, called Memory-Inductive Categorial Grammar (MICG) (Boonkwan and Supnithi, 2008). Fillers and gaps, encoded as memory modalities, are modified to syntactic categories, and they are associated by the gap-resolution connective when coordination and serialization take place. Though their framework is successful in resolving a wide variety of gapping, its generative power falls between LIG and Indexed Grammar, theoretically too powerful for natural languages.

The memory mechanism introduced in this paper deals with fillers and gaps in SVC. It is similar to anaphoric resolution in ATN, Jacobson's model, TLG, and CTL. However, it also has prominent distinction from them: The anaphoric mechanisms mentioned earlier are dealing with unbounded dependency or even inter-sentential ellipses, while the memory mechanism in this paper is dealing only with intra-sentential bounded dependency in SVC as generalized in (13). Moreover, choices of filler-gap association can be pruned out by the use of combinatory directionality because the word order of analytic languages is fixed. It is noticeable that we can simply determine the grammatical function (subject or object) of arbitrary np's in (13) from the directionality (the subject on the left and the object on the right). With these reasons, I therefore adapted the notions of MICG's memory modalities and gap-resolution connective (Boonkwan and Supnithi, 2008) for the backbone of the memory mechanism.

In CCG with Memory Mechanism (CCG-MM), syntactic categories are modalized with memory modalities. For each functional application, a syntactic category can be stored, or *memorized*, into the filler storage and the resulted category is modalized with the filler $\square$. A syntactic category can also be induced as a gap in a unary derivation called *induction* and the resulted category is modalized with the gap $\diamond$.

There are two constraint parameters in each modality: the combinatory directionality $d \in \{<, >\}$ and the syntactic category c, resulting in the filler and the gap denoted in the forms $\square_c^d$ and $\diamond_c^d$, respectively. For example, the syntactic category $\square_{\mathtt{np}}^{<}\diamond_{\mathtt{np}}^{>}\mathtt{s}$ has a filler of type np on the left side and a gap of type np on the right side.

The filler $\square_c^d$ and the gap $\diamond_c^d$ of the same directionality and syntactic categories are said to be *symmetric* under the gap-resolution connective $\oplus$; that is, they are matched and canceled in the gap resolution process. Apart from MICG, I restrict the associative power of $\oplus$ to match only a filler and a gap, not between two gaps, so that the generative power can be preserved linear. This topic will be discussed in §5. Given two strings of modalities m_1 and m_2, the gap-resolution connective $\oplus$ is defined in (21).

$$\begin{aligned} \square_c^d m_1 \oplus \diamond_c^d m_2 &\equiv m_1 \oplus m_2 \\ \diamond_c^d m_1 \oplus \square_c^d m_2 &\equiv m_1 \oplus m_2 \\ \epsilon \oplus \epsilon &\equiv \epsilon \end{aligned} \tag{21}$$

The notation ϵ denotes an empty string. It means that a syntactic category modalized with an empty modality string is simply *unmodalized*; that is, any modalized syntactic categories $\epsilon\mathtt{X}$ are equivalent to the unmodalized ones X.

Since the syntactic categories are modalized by a modality string, all combinatory operations in canonical CCG must preserve the modalities after each derivation step. However, there are two conditions to be satisfied:

Condition A: At least one operands of functional application must be unmodalized.

Condition B: Both operands of functional composition, disharmonic functional composition, and type raising must be unmodalized.

Both conditions are introduced to preserve the generative power of CCG. This topic will be discussed in §5.

As adopted from MICG, there are two memory operations: memorization and induction.

Memorization: a filler modality is pushed to the top of the memory when an functional application rule is applied, where the filler's syntactic category must be unmodalized. Let m be a modal-

ity string, the memorization operation is defined in (22).

(22)
$$\begin{array}{llll} \epsilon\mathtt{X/Y} \quad m\mathtt{Y} & \Rightarrow & \Box^{<}_{\mathtt{X/Y}} m\mathtt{X} & [> \mathbf{M}_F] \\ m\mathtt{X/Y} \quad \epsilon\mathtt{Y} & \Rightarrow & \Box^{>}_{\mathtt{Y}} m\mathtt{X} & [> \mathbf{M}_A] \\ \epsilon\mathtt{Y} \quad m\mathtt{X\backslash Y} & \Rightarrow & \Box^{<}_{\mathtt{Y}} m\mathtt{X} & [< \mathbf{M}_A] \\ m\mathtt{Y} \quad \epsilon\mathtt{X\backslash Y} & \Rightarrow & \Box^{>}_{\mathtt{X\backslash Y}} m\mathtt{X} & [< \mathbf{M}_F] \end{array}$$

Induction: a gap modality is pushed to the top of the memory when a gap of such type is induced at either side of the syntactic category. Let m be a modality string, the induction operation is defined in (23).

(23)
$$\begin{array}{rcll} m\mathtt{X/Y} & \Rightarrow & \Diamond^{>}_{\mathtt{Y}} m\mathtt{X} & [> \mathbf{I}_A] \\ m\mathtt{Y} & \Rightarrow & \Diamond^{<}_{\mathtt{X/Y}} m\mathtt{X} & [> \mathbf{I}_F] \\ m\mathtt{X\backslash Y} & \Rightarrow & \Diamond^{<}_{\mathtt{Y}} m\mathtt{X} & [< \mathbf{I}_A] \\ m\mathtt{Y} & \Rightarrow & \Diamond^{>}_{\mathtt{X\backslash Y}} m\mathtt{X} & [< \mathbf{I}_F] \end{array}$$

Because the use of memory mechanism elucidates fillers and gaps hidden in the derivation, we can then replace the decomposition rule of the canonical CCG with the gap resolution process of MICG. Fillers and gaps are associated in the coordination and serialization by the gap-resolution connective $\oplus$. For any given m_1, m_2, if $m_1 \oplus m_2$ exists then always $m_1 \oplus m_2 \equiv \epsilon$. Given two modality strings m_1 and m_2 such that $m_1 \oplus m_2$ exists, the coordination rule (Φ) and serialization rule (Σ) are redefined on $\oplus$ in (24).

(24)
$$\begin{array}{llllll} m_1\mathtt{X} & \& & m_2\mathtt{X} & \Rightarrow & \mathtt{X} & [\mathbf{\Phi}] \\ m_1\mathtt{X} & & m_2\mathtt{X} & \Rightarrow & \mathtt{X} & [\mathbf{\Sigma}] \end{array}$$

At present, the memory mechanism was developed in Prolog for the sake of unification mechanism. Each induction rule is nondeterministically applied and variables are sometimes left uninstantiated. For example, the sentence in (12) can be parsed as illustrated in (25).

(25)
$$\dfrac{\dfrac{\mathtt{John} \quad \dfrac{\dfrac{\mathtt{eats} \quad \mathtt{sandwiches}}{\mathtt{s\backslash np/np} \quad \mathtt{np}}}{\Box^{<}_{\mathtt{s\backslash np/np}}\mathtt{s\backslash np}}\,{>}\mathbf{M}_F \quad \mathtt{and} \quad \dfrac{\dfrac{\mathtt{Mary} \quad \dfrac{\mathtt{noodles}}{\mathtt{np}}}{\Diamond^{<}_{X_1/\mathtt{np}} X_1}\,{>}\mathbf{I}_F}{\Diamond^{<}_{X_2\backslash\mathtt{np}/\mathtt{np}} X_2}\,{<}}{\Box^{<}_{\mathtt{s\backslash np/np}}\mathtt{s}}\,{<}}{\mathtt{s}}\,\Phi$$

Let us consider the derivation in the right conjunct. The gap induction is first applied on np resulting in $\Diamond^{<}_{X_1/\mathtt{np}} X_1$, where X_1 is an uninstantiated variable. Then the backward application is applied, so that X_1 is unified with $X_2\backslash\mathtt{np}$. Finally, the left and the right conjuncts are coordinated yielding that X_2 is unified with s and X_1 with $\mathtt{s\backslash np}$. For convenience of type-setting, let us suppose that we can always choose the right type in each induction step and suppress the unification process.

Table 1: Slash modalities for memory operations.

	- Left	+ Left
- Right	$\star$	$\triangleleft$
+ Right	$\triangleright$	$\cdot$

Once we instantiate X_1 and X_2, the derivation obtained in (25) is quite more straightforward than the derivation in (12). The filler *eats* is introduced on the left conjunct, while the gap of type $\mathtt{s\backslash np/np}$ is induced on the right conjunct. The coordination operation associates the filler and the gap resulting in a complete derivation.

A significant feature of the memory mechanism is that it handles all kinds of intra-sentential ellipses in SVC. This is because the coordination and serialization rules allow pro-dropping in either the left or the right conjunct. For example, the intra-sentential ellipses pattern in Thai SVC illustrated in (19) can be derived as illustrated in (26).

(26)
$$\dfrac{\dfrac{\mathtt{Kla} \quad \dfrac{\dfrac{\mathtt{go\ follow\ seek\ in\ cane\text{-}field} \quad \dfrac{\mathtt{find} \quad \dfrac{\mathtt{Laay} \quad \dfrac{\mathtt{FUT\ walk\ leave\ go}}{\mathtt{s\backslash np}}}{\mathtt{np}}}{\mathtt{s\backslash np/(s\backslash np)/np}}}{\mathtt{s\backslash np/np}}\,{>}\mathbf{M}_A}{\Diamond^{>}_{\mathtt{np}}\mathtt{s\backslash np}}\,{>}\mathbf{I}_A \quad \dfrac{}{\Box^{>}_{\mathtt{np}}\mathtt{s\backslash np/(s\backslash np)}}\,{>}}{\Box^{>}_{\mathtt{np}}\mathtt{s\backslash np}}\,{>}}{\mathtt{s\backslash np}}\,\Sigma}{\mathtt{s}}\,{<}$$

By replacing the decomposition rule with the memory mechanism, CCG accepts all patterns of pro-dropping in SVC. It should also be noted that the derivation in (20) is *per se* prohibited by the coordination rule.

Similar to canonical CCG, CCG-MM is also *resource-sensitive*; that is, each combinatory operation is allowed or prohibited with respect to the resource we have (Baldridge and Kruijff, 2003). Baldridge (2002) showed that we can obtain a cleaner resource management in canonical CCG by the use of modalized slashes to control combinatory behavior. His multimodal schema of slash permissions can also be applied to the memory mechanism in much the same way. I assume that there are four modes of memory operations according to direction and allowance of memory operations as in Table 1.

The modes can be organized into the type hierarchy shown in Figure 1. The slash modality $\star$, the most limited mode, does not allow any memory operations on both sides. The slash modalities $\triangleleft$ and $\triangleright$ allow memorization and induction on the

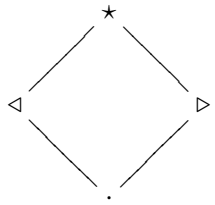


Figure 1: Hierarchy of slash modalities for memory operations.

left and right sides, respectively. Finally, the slash modality $\cdot$ allows memorization and induction on both sides. In order to distinguish the memory operation's slash modalities from Baldridge's slash modalities, I annotate the first as a superscript and the second as a subscript of the slashes. For example, the syntactic category $\mathtt{s}\backslash^{\triangleleft}_{\times}\mathtt{np}$ denotes that $\mathtt{s}\backslash\mathtt{np}$ allows permutation in crossed functional composition ($\times$) and memory operations on the left side ($\triangleleft$). As with Baldridge's multimodal framework, the slash modality $\cdot$ can be omitted from writing. By defining the slash modalities, it follows that the memory operations can be defined in (27).

(27)
$$\begin{array}{rcl@{\quad}l}
m\mathtt{X}/^{\triangleright}\mathtt{Y} \;\; \epsilon\mathtt{Y} & \Rightarrow & \Box^{>}_{\mathtt{Y}} m\mathtt{X} & [> \mathbf{M}_F] \\
\epsilon\mathtt{X}/^{\triangleleft}\mathtt{Y} \;\; m\mathtt{Y} & \Rightarrow & \Box^{<}_{\mathtt{X}/^{\triangleleft}\mathtt{Y}} m\mathtt{X} & [> \mathbf{M}_A] \\
\epsilon\mathtt{Y} \;\; m\mathtt{X}\backslash^{\triangleleft}\mathtt{Y} & \Rightarrow & \Box^{<}_{\mathtt{Y}} m\mathtt{X} & [< \mathbf{M}_A] \\
m\mathtt{Y} \;\; \epsilon\mathtt{X}\backslash^{\triangleright}\mathtt{Y} & \Rightarrow & \Box^{>}_{\mathtt{X}\backslash^{\triangleright}\mathtt{Y}} m\mathtt{X} & [< \mathbf{M}_F] \\
m\mathtt{X}/^{\triangleright}\mathtt{Y} & \Rightarrow & \Diamond^{>}_{\mathtt{Y}} m\mathtt{X} & [> \mathbf{I}_A] \\
m\mathtt{Y} & \Rightarrow & \Diamond^{<}_{\mathtt{X}/^{\triangleleft}\mathtt{Y}} m\mathtt{X} & [> \mathbf{I}_F] \\
m\mathtt{X}\backslash^{\triangleleft}\mathtt{Y} & \Rightarrow & \Diamond^{<}_{\mathtt{Y}} m\mathtt{X} & [< \mathbf{I}_A] \\
m\mathtt{Y} & \Rightarrow & \Diamond^{>}_{\mathtt{X}\backslash^{\triangleright}\mathtt{Y}} m\mathtt{X} & [< \mathbf{I}_F]
\end{array}$$

When incorporating with the memory mechanism and the slash modalities, CCG becomes flexible enough to handle all patterns of intra-sentential ellipses in SVC which are prevalent in analytic languages, and to manage its lexical resource. I will now show that CCG-MM extends the generative power of the canonical CCG.

5 Generative Power

In this section, we will informally discuss the margin of generative power introduced by the memory mechanism. Since Vijay-Shanker (1994) showed that CCG and Linear Indexed Grammar (LIG) (Gazdar, 1988) are weakly equivalent; i.e. they generate the same sets of strings, we will first compare the CCG-MM with the LIG. As will be shown, its generative power is beyond LIG; we will find the closest upper bound in order to locate it in the Chomsky's hierarchy.

We will follow the equivalent proof of Vijay-Shanker and Weir (1994) to investigate the generative power of CCG-MM. Let us first assume that we are going to construct an LIG $G = (V_N, V_T, V_S, S, P)$ that subsumes CCG-MM. To construct G, let us define each of its component as follows.

> V_N is a finite set of syntactic categories,
> V_T is a finite set of terminals,
> V_S is a finite set of stack symbols having the form $\Box^d_c$, $\Diamond^d_c$, $/c$, or $\backslash c$,
> $S \in V_N$ is the start symbol, and
> P is a finite set of productions, having the form
>
> $$\begin{array}{rcl} A[] & \rightarrow & a \\ A[\circ \circ l] & \rightarrow & A_1[] \ldots A_i[\circ \circ l'] \ldots A_n[] \end{array}$$
>
> where each $A_k \in V_N$, $d \in \{<, >\}$, $c \in V_N$, $l, l' \in V_S$, and $a \in V_T \cup \{\epsilon\}$.

The notation for stacks uses $[\circ \circ l]$ to denote an arbitrary stack whose top symbol is l. The *linearity* of LIG comes from the fact that in each production there is only one daughter that share the stack features with its mother. Let us also define $\Delta(\sigma)$ as the homomorphic function that converts each modality in a modality string σ into its symmetric counterpart, i.e. a filler $\Box^d_c$ into a gap $\Diamond^d_c$, and vice versa. The stack in this LIG is used for storing (1) tailing slashes of a syntactic category for harmonic/disharmonic functional composition rules, and (2) modalities of a syntactic category for gap resolution.

We start out by transforming the lexical item. For every lexical item of the form $w \vdash \mathtt{X}$ where $\mathtt{X}$ is a syntactic category, add the following production to P:

(28)
$$\mathtt{X}[] \;\rightarrow\; w$$

We add two unary rules for converting between tailing slashes and stack values. For every syntactic category $\mathtt{X}$ and $\mathtt{Y_1}, \ldots, \mathtt{Y_n}$, the following rules are added.

(29)
$$\begin{array}{rcl} \mathtt{X}|_1\mathtt{Y}_1 \ldots |_n\mathtt{Y}_n[\circ\circ] & \rightarrow & \mathtt{X}[\circ \circ |_1\mathtt{Y}_1 \ldots |_n\mathtt{Y}_n] \\ \mathtt{X}[\circ \circ |_1\mathtt{Y}_1 \ldots |_n\mathtt{Y}_n] & \rightarrow & \mathtt{X}|_1\mathtt{Y}_1 \ldots |_n\mathtt{Y}_n[\circ\circ] \end{array}$$

where the top of $\circ\circ$ must be a filler or a gap, or $\circ\circ$ must be empty. This constraint preserves the ordering of combinatory operations.

We then transform the functional application rules into LIG productions. From Condition A, we can generalize the functional application rules in (2) as follows.

(30)
$$\begin{array}{llll} m\mathtt{X}/\mathtt{Y} & \mathtt{Y} & \Rightarrow & m\mathtt{X} \\ \mathtt{X}/\mathtt{Y} & m\mathtt{Y} & \Rightarrow & m\mathtt{X} \\ m\mathtt{Y} & \mathtt{X}\backslash\mathtt{Y} & \Rightarrow & m\mathtt{X} \\ \mathtt{Y} & m\mathtt{X}\backslash\mathtt{Y} & \Rightarrow & m\mathtt{X} \end{array}$$

where m is a modality string. Condition A preserves the linearity of the generative power in that it prevents the functional application rules from involving the two stacks of the daughters at once. We can convert the rules in (30) into the following productions.

(31)
$$\begin{array}{lcll} \mathtt{X}[\circ\circ] & \rightarrow & \mathtt{X}[\circ\circ/\mathtt{Y}] & \mathtt{Y}[] \\ \mathtt{X}[\circ\circ] & \rightarrow & \mathtt{X}[/\mathtt{Y}] & \mathtt{Y}[\circ\circ] \\ \mathtt{X}[\circ\circ] & \rightarrow & \mathtt{Y}[\circ\circ] & \mathtt{X}[\backslash\mathtt{Y}] \\ \mathtt{X}[\circ\circ] & \rightarrow & \mathtt{Y}[] & \mathtt{X}[\circ\circ\backslash\mathtt{Y}] \end{array}$$

We can generalize the harmonic and disharmonic, forward and backward composition rules in (6) and (9) as follows.

(32)
$$\begin{array}{llll} \mathtt{X}/\mathtt{Y} & \mathtt{Y}|_1\mathtt{Z}_1\ldots|_n\mathtt{Z}_n & \Rightarrow & \mathtt{X}|_1\mathtt{Z}_1\ldots|_n\mathtt{Z}_n \\ \mathtt{Y}|_1\mathtt{Z}_1\ldots|_n\mathtt{Z}_n & \mathtt{X}\backslash\mathtt{Y} & \Rightarrow & \mathtt{X}|_1\mathtt{Z}_1\ldots|_n\mathtt{Z}_n \end{array}$$

where each $|_i \in \{\backslash, /\}$. By Condition B, we obtain that all operands are unmodalized so that we can treat only tailing slashes. That is, Condition B prevents us from processing both tailing slashes and memory modalities at once where the linearity of the rules is deteriorated. We can therefore convert these rules into the following productions.

(33)
$$\begin{array}{lcll} \mathtt{X}[\circ\circ] & \rightarrow & \mathtt{X}[/\mathtt{Y}] & \mathtt{Y}[\circ\circ] \\ \mathtt{X}[\circ\circ] & \rightarrow & \mathtt{Y}[\circ\circ] & \mathtt{X}[\backslash\mathtt{Y}] \end{array}$$

The memorization and induction rules described in (27) are transformed into the following productions.

(34)
$$\begin{array}{lcll} \mathtt{X}[\circ\circ\square^{<}_{\mathtt{X}/\mathtt{Y}}] & \rightarrow & \mathtt{X}[/\mathtt{Y}] & \mathtt{Y}[\circ\circ] \\ \mathtt{X}[\circ\circ\square^{>}_{\mathtt{Y}}] & \rightarrow & \mathtt{X}[\circ\circ/\mathtt{Y}] & \mathtt{Y}[] \\ \mathtt{X}[\circ\circ\square^{<}_{\mathtt{Y}}] & \rightarrow & \mathtt{Y}[] & \mathtt{X}[\circ\circ\backslash\mathtt{Y}] \\ \mathtt{X}[\circ\circ\square^{>}_{\mathtt{X}\backslash\mathtt{Y}}] & \rightarrow & \mathtt{Y}[\circ\circ] & \mathtt{X}[\backslash\mathtt{Y}] \\ \mathtt{X}[\circ\circ\diamond^{>}_{\mathtt{Y}}] & \rightarrow & \mathtt{X}[\circ\circ/\mathtt{Y}] & \\ \mathtt{X}[\circ\circ\diamond^{<}_{\mathtt{X}/\mathtt{Y}}] & \rightarrow & \mathtt{Y}[\circ\circ] & \\ \mathtt{X}[\circ\circ\diamond^{<}_{\mathtt{Y}}] & \rightarrow & \mathtt{X}[\circ\circ\backslash\mathtt{Y}] & \\ \mathtt{X}[\circ\circ\diamond^{>}_{\mathtt{X}\backslash\mathtt{Y}}] & \rightarrow & \mathtt{Y}[\circ\circ] & \end{array}$$

However, it is important to take into account the coordination and serialization rules, because they involve two stacks which have similar stack values if we convert one of them into the symmetric form with Δ. Those rules can be transformed as follows.

(35)
$$\begin{array}{lcllll} \mathtt{X}[] & \rightarrow & \mathtt{X}[\circ\circ] & \&[] & \mathtt{X}[\Delta(\circ\circ)] \\ \mathtt{X}[] & \rightarrow & \mathtt{X}[\circ\circ] & \mathtt{X}[\Delta(\circ\circ)] & \end{array}$$

It is obvious that the rules in (35) are not LIG production; that is, CCG-MM cannot be generated by any LIGs; or more precisely, CCG-MM is properly more powerful than CCG. We therefore have to find an upper bound of its generative power.

Though CCG-MM is more powerful than CCG and LIG, the rules in (35) reveal a significant property of Partially Linear Indexed Grammar (PLIG) (Keller and Weir, 1995), an extension of LIG whose productions are allowed to have two or more daughters sharing stack features with each other but these stacks are not shared with their mother as shown in (36).

(36)
$$A[] \rightarrow A_1[]\ldots A_i[\circ\circ]\ldots A_j[\circ\circ]\ldots A_n[]$$

Whereby restricting the power of the gap-resolution connective, the two stacks of the daughters are shared but not with their mother. An interesting trait of PLIG is that it can generate the language $\{w^k | w$ is in a regular language and $k \in \mathcal{N}\}$. This is similar to the pattern of SVC in which a series of verb phrase can be reduplicated.

To conclude this section, CCG-MM is more powerful than LIG but less powerful than PLIG. From (Keller and Weir, 1995), we can position the CCG-MM in the Chomsky's hierarchy as follows: CFG $<$ CCG = TAG = HG = LIG $<$ CCG-MM $\leq$ PLIG $\leq$ LCFRS $<$ CSG.

6 Conclusion and Future Work

I have presented an approach to treating serial verb construction in analytic languages by incorporating CCG with a memory mechanism. In the memory mechanism, fillers and gaps are stored as modalities that modalize a syntactic category. The fillers and the gaps are then associated in the coordination and the serialization rules. This results in a more flexible way of dealing with intra-sentential ellipses in SVC than the decomposition rule in canonical CCG. Theoretically speaking, the proposed memory mechanism increases the generative power of CCG into the class of partially linear indexed grammars.

Future research remains as follows. First, I will investigate constraints that reduce the search space of parsing caused by gap induction. Second, I will apply the memory mechanism in solving discontinuous gaps. Third, I will then extend this framework to free word-ordered languages. Fourth and finally, the future direction of this research is to develop a wide-coverage parser in which statistics is also made use to predict memory operations occuring in derivation.

References

Jason Baldridge and Geert-Jan M. Kruijff. 2003. Multimodal combinatory categorial grammar. In *Proceedings of the 10th Conference of the European Chapter of the ACL 2003*, pages 211–218, Budapest, Hungary.

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

Prachya Boonkwan and Thepchai Supnithi. 2008. Memory-inductive categorial grammar: An approach to gap resolution in analytic-language translation. In *Proceedings of The Third International Joint Conference on Natural Language Processing*, volume 1, pages 80–87, Hyderabad, India, January.

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

Petra Hendriks. 1995. Ellipsis and multimodal categorial type logic. In *Proceedings of Formal Grammar Conference*, pages 107–122. Barcelona, Spain.

Pauline Jacobson. 1999. Towards a variable-free semantics. *Linguistics and Philosophy*, 22:117–184, October.

Gerhard Jäger. 1997. Anaphora and ellipsis in type-logical grammar. In *Proceedings of the 11th Amsterdam Colloquium*, pages 175–180, Amsterdam, the Netherland. ILLC, Universiteit van Amsterdam.

Gerhard Jäger. 2001. Anaphora and quantification in categorial grammar. In *Lecture Notes in Computer Science; Selected papers from the 3rd International Conference, on logical aspects of Computational Linguistics*, volume 2014/2001, pages 70–89.

Bill Keller and David Weir. 1995. A tractable extension of linear indexed grammars. In *In Proceedings of the 7th European Chapter of ACL Conference*.

Charles N. Li and Sandra A. Thompson. 1981. *Mandarin Chinese: A Functional Reference Grammar*. Berkeley: University of California Press.

Michael Moortgat. 1997. Categorial type logics. In van Benthem and ter Meulen, editors, *Handbook of Logic and Language*, chapter 2, pages 163–170. Elsevier/MIT Press.

Glyn Morrill. 1994. Type logical grammar. In *Categorial Logic of Signs*. Kluwer, Dordrecht.

Nuttanart Muansuwan. 2002. *Verb Complexes in Thai*. Ph.D. thesis, University at Buffalo, The State University of New York.

Richard T. Oehrle, 2007. *Non-Transformational Syntax: A Guide to Current Models*, chapter Multimodal Type Logical Grammar. Oxford: Blackwell.

Mark Steedman. 1990. Gapping as constituent coordination. *Linguistics and Philosophy*, 13:207–263.

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

Kingkarn Thepkanjana. 1986. *Serial Verb Constructions in Thai*. Ph.D. thesis, University of Michigan.

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

William A. Woods. 1970. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, October.