14

Linearization of affine abstract categorial grammars

Ryo Yoshinaka

Abstract

The abstract categorial grammar (ACG) is a grammar formalism based on linear lambda calculus. It is natural to ask how the expressive power of ACGs increases when we relax the linearity constraint on the formalism. This paper introduces the notion of affine ACGs by extending the definition of original ACGs, and presents a procedure for converting a given affine ACG into a linear ACG whose language is exactly the set of linear λ-terms generated by the original affine ACG.

Keywords Abstract categorial grammars, generative capacity, lambda calculus, context-free tree grammars, linear context-free rewriting systems, multiple context-free grammars

14.1 Introduction

De Groote (2001) has introduced *abstract categorial grammars (ACGs)*, in which both *lexical entries* of the grammar as well as *grammatical combinations* of them are represented by simply typed linear λ-terms. While the linearity constraint on grammatical combinations is thought to be reasonable, admitting non-linear λ-terms as lexical entries may allow ACGs to describe linguistic phenomena in a more natural and concise fashion.

On the other hand, de Groote and Pogodalla (2003, 2004) have shown that a variety of context-free formalisms, namely, context-free grammars, linear[1] context-free tree grammars (linear CFTGs)[2] and linear context-free rewrit-

[1] This paper lets the term "linearity" mean non-duplication and non-deletion. Thus "linear CFTGs" means non-duplicating non-deleting CFTGs here, though usually "linear CFTGs" means non-duplicating CFTGs.

[2] See also Kanazawa and Yoshinaka (2005) for complete proof of encodability of linear

Proceedings of FG-2006.
Editor: Shuly Wintner.

ing systems (LCFRSs), is encoded by ACGs in straightforward ways. In this sense, ACGs can be thought of as a generalization of those grammar formalisms. The linearity constraint in those formalisms matches that of the ACG formalism.

Concerning those grammar formalisms, it is known that the expressive power does not change when the linearity constraint is relaxed to just non-duplication, allowing deleting operations. Seki et al. (1991) have shown the equivalence between LCFRSs and multiple context-free grammars (MCFGs), which correspond to the relaxed version of LCFRSs that may have deleting operations. Fujiyoshi (2005) has established the equivalence between linear monadic CFTGs and non-duplicating monadic CFTGs. Fisher's result (Fisher, 1968a,b) is rather general. He has shown that the string IO-languages generated by general CFTGs coincide with the string IO-languages generated by non-deleting CFTGs.

Along this line, extending the definition of usual linear ACGs, this paper introduces *affine ACGs*, which have BCKλ-terms as their lexical entries, and compares the generative power of linear ACGs and affine ACGs. We present a procedure for converting a given affine ACG into a linear ACG whose language is exactly the set of the linear λ-terms generated by the original ACG. Therefore, affine ACGs are not essentially more expressive than linear ACGs, since strings and trees are usually represented with linear λ-terms.

As linear ACGs encode linear CFTGs and LCFRSs, affine ACGs encode non-duplicating CFTGs and MCFGs in straightforward ways. For such affine ACGs, our linearization method constructs linear ACGs which have the form corresponding to linear CFTGs or LCFRSs. Thus, our result is a generalization of the results we have mentioned above with the exception of Fisher's, which covers CFTGs involving duplication.

14.2 Preliminaries

14.2.1 Lambda-Terms

Let $\mathscr{A}$ be a finite non-empty set of *atomic types*. The set $\mathcal{T}(\mathscr{A})$ of *types* built on $\mathscr{A}$ is defined as the smallest superset of $\mathscr{A}$ such that

- if $\alpha, \beta \in \mathcal{T}(\mathscr{A})$, then $(\alpha \to \beta) \in \mathcal{T}(\mathscr{A})$.

The *order* of a type is given by the function ord : $\mathcal{T}(\mathscr{A}) \to \mathbb{N}$,

- $\mathrm{ord}(p) = 1$ for all $p \in \mathscr{A}$,
- $\mathrm{ord}((\alpha \to \beta)) = \max\{\mathrm{ord}(\alpha) + 1, \mathrm{ord}(\beta)\}$.

A *higher-order signature* Σ is a triple $\langle \mathscr{A}, \mathscr{C}, \tau \rangle$ where $\mathscr{A}$ is a finite non-empty set of atomic types, $\mathscr{C}$ is a finite set of constants, and τ is a func-

CFTGs by ACGs.

tion from $\mathscr{C}$ to $\mathcal{T}(\mathscr{A})$. The *order* of the higher-order signature is defined as $\mathrm{ord}(\Sigma) = \max\{\, \mathrm{ord}(\tau(\mathsf{a})) \mid \mathsf{a} \in \mathscr{C} \,\}$.

Let $\mathscr{X}$ be a countably infinite set of *variables*. The set $\Lambda(\Sigma)$ of λ-*terms* (*terms* for short) built upon Σ and the type $\hat{\tau}(M)$ of a term $M \in \Lambda(\Sigma)$ are defined inductively as follows:

- For every $\mathsf{a} \in \mathscr{C}$, $\mathsf{a} \in \Lambda(\Sigma)$ and $\hat{\tau}(\mathsf{a}) = \tau(\mathsf{a})$.
- For every $x \in \mathscr{X}$ and $\alpha \in \mathcal{T}(\mathscr{A})$, $x^\alpha \in \Lambda(\Sigma)$ and $\hat{\tau}(x^\alpha) = \alpha$.
- For $M, N \in \Lambda(\Sigma)$, if $\hat{\tau}(M) = (\alpha \to \beta)$, $\hat{\tau}(N) = \alpha$, then $(MN) \in \Lambda(\Sigma)$ and $\hat{\tau}((MN)) = \beta$.
- For $x \in \mathscr{X}$, $\alpha \in \mathcal{T}(\mathscr{A})$ and $M \in \Lambda(\Sigma)$, $(\lambda x^\alpha.M) \in \Lambda(\Sigma)$ and $\hat{\tau}((\lambda x^\alpha.M)) = (\alpha \to \hat{\tau}(M))$.

For convenience, we simply write τ instead of $\hat{\tau}$ and often omit the superscript on a variable if its type is clear from the context. The notions of free variables, closed terms, β-normal form, $\beta\eta$-normal form, are defined as usual (see Hindley (1997) for instance). A term M is a *combinator* iff M is closed and M contains no constants. A term M is said to be *affine* if any variable occurs free at most once in every sub-term of M. An affine term is said to be *linear* if every λ-abstraction binds exactly one occurrence of a variable. The sets of affine and linear terms are respectively denoted by $\Lambda^{\mathrm{aff}}(\Sigma)$ and $\Lambda^{\mathrm{lin}}(\Sigma)$. As usual, let $\twoheadrightarrow_\beta$, $=_\beta$, $=_{\beta\eta}$, $\equiv$ denote β-reduction, β-equality, $\beta\eta$-equality, and α-equivalence respectively. $|M|_\beta$ and $|M|_{\beta\eta}$ respectively represent the β-normal form and $\beta\eta$-normal form. We use upper case italic letters $M, N, P, \ldots$ for terms, late lower case italic letters $x, y, z, \ldots$ for variables, middle lower case italic letters $o, p, \ldots$ for atomic types, Greek letters $\alpha, \beta, \ldots$ for types, sanserif $\mathsf{a}, \mathsf{A}, \ldots$ for constants. We write $\alpha \to \beta \to \gamma \to \delta$ for $(\alpha \to (\beta \to (\gamma \to \delta)))$, $\alpha^3 \to \delta$ for $\alpha \to \alpha \to \alpha \to \delta$, $MNPQ$ for $(((MN)P)Q)$, $\lambda xyz.M$ for $(\lambda x.(\lambda y.(\lambda z.M)))$, and so on.

14.2.2 Abstract Categorial Grammars

Definition 12 For two sets of atomic types $\mathscr{A}_0$ and $\mathscr{A}_1$, a *type substitution* σ is a mapping from $\mathscr{A}_0$ to $\mathcal{T}(\mathscr{A}_1)$, which can be extended homomorphically as

$$\sigma(\alpha \to \beta) = \sigma(\alpha) \to \sigma(\beta).$$

For two higher-order signatures Σ_0 and Σ_1, a *term substitution* θ is a mapping from $\mathscr{C}_0$ to $\Lambda(\Sigma_1)$ such that $\theta(\mathsf{a})$ is closed for all $\mathsf{a} \in \mathscr{C}_0$. For two higher-order signatures Σ_0 and Σ_1, we say that a type substitution $\sigma : \mathscr{A}_0 \to \mathcal{T}(\mathscr{A}_1)$ and a term substitution $\theta : \mathscr{C}_0 \to \Lambda(\Sigma_1)$ are *compatible* iff $\sigma(\tau_0(\mathsf{a})) = \tau_1(\theta(\mathsf{a}))$ holds for all $\mathsf{a} \in \mathscr{C}_0$. A *lexicon* from Σ_0 to Σ_1 is a compatible pair of a type substitution and a term substitution. A lexicon $\mathscr{L} = \langle \sigma, \theta \rangle$ is *affine* (*linear*) iff $\theta(\mathsf{a})$ is affine (linear) for all $\mathsf{a} \in \mathscr{C}_0$. For a lexicon $\mathscr{L} = \langle \sigma, \theta \rangle$, we define $\hat{\theta}$ as the homomorphic extension of θ such that $\hat{\theta}(x^\alpha) = x^{\sigma(\alpha)}$. Indeed, $\hat{\theta}(M)$ is

always a well-typed λ-term if so is M; if M has type α, then $\hat{\theta}(M)$ has type $\sigma(\alpha)$.

Hereafter we identify a lexicon $\mathscr{L} = \langle \sigma, \theta \rangle$ with the functions σ and $\hat{\theta}$. A lexicon $\mathscr{L}$ is *n-th order* if $\mathrm{ord}(\mathscr{L}) = \max\{ \mathrm{ord}(\sigma(p)) \mid p \in \mathscr{A}_0 \} \leq n$.

Definition 13 An *abstract categorial grammar (ACG)* is a quadruple $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$, where

- Σ_0 is a higher-order signature, called the *abstract vocabulary*,
- Σ_1 is a higher-order signature, called the *object vocabulary*,
- $\mathscr{L}$ is a linear lexicon from Σ_0 to Σ_1,
- $s \in \mathscr{A}_0$ is called the *distinguished type*.

We sometimes call the triple $\langle \mathsf{a}, \tau_0(\mathsf{a}), \mathscr{L}(\mathsf{a}) \rangle$ for $\mathsf{a} \in \mathscr{C}_0$ a *lexical entry*, and specify an ACG by giving the set of lexical entries and the distinguished type.

Definition 14 An ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$ generates two languages, the *abstract language* $\mathcal{A}(\mathscr{G})$ and the *object language* $\mathcal{O}(\mathscr{G})$, defined as

$$\mathcal{A}(\mathscr{G}) = \{ M \mid M \in \Lambda^{\mathrm{lin}}(\Sigma_0) \text{ is a closed } \beta\eta\text{-normal term of type } s \},$$
$$\mathcal{O}(\mathscr{G}) = \{ |\mathscr{L}(M)|_{\beta\eta} \mid M \in \mathcal{A}(\mathscr{G}) \}.$$

The abstract language can be thought of as a set of abstract grammatical structures, and the object language is regarded as the set of concrete forms obtained from these abstract structures and the lexicon. Thus, we simply say the language generated by an ACG for its object language. The term *abstract categorial languages (ACLs)* means the object languages of ACGs.

Though de Groote's original definition of an ACG requires the lexicon to be linear, this paper allows the lexicon to be non-linear. We call an ACG whose lexicon is affine *affine ACG*, and denote the class of affine ACGs by $\mathbf{G}^{\mathrm{aff}}$. We then distinguish affine ACGs whose lexicons are linear, i.e., original ACGs, by calling them *linear ACGs* and let $\mathbf{G}^{\mathrm{lin}}$ denote the class of linear ACGs. Note that the abstract language always consists of *linear* terms, though an ACG is not necessarily linear. For each $\mathbf{G}^* \in \{\mathbf{G}^{\mathrm{lin}}, \mathbf{G}^{\mathrm{aff}}\}$, $\mathbf{G}^*(m, n)$ denotes the subclass of ACGs belonging to $\mathbf{G}^*$ such that the order of the abstract vocabulary is at most m and the order of the lexicon is at most n. An ACG is *m-th order* if it belongs to $\mathbf{G}^*(m, n)$ for some n.

Example 1 Let $\mathit{str} = o \to o$ and $M + N$ be an abbreviation of $\lambda z^o.M(Nz)$ if the types of M and N are str. Let us consider the affine ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$

with the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}(\mathsf{x})$
C	n	$\lambda v.v/\mathsf{cat}//\mathsf{cats}/$
M	n	$\lambda v.v/\mathsf{mouse}//\mathsf{mice}/$
J	np	$\lambda y.y/\mathsf{John}/P_1$
R	$np \to s$	$\lambda x.x(\lambda uv.u + v/\mathsf{runs}//\mathsf{run}/)$
E	$np^2 \to s$	$\lambda x_1 x_2.x_2(\lambda uv.u + v/\mathsf{eats}//\mathsf{eat}/) + x_1(\lambda uv.u)$
A	$n \to np$	$\lambda zy.y(/\mathsf{a}/ + zP_1)P_1$
L	$n \to np$	$\lambda zy.y(/\mathsf{all}/ + zP_2)P_2$

where each /xxx/ is a constant of type str, P_i denotes $\lambda u_1^{\mathit{str}} u_2^{\mathit{str}}.u_i$, $\mathscr{L}(n) = (\mathit{str}^2 \to \mathit{str}) \to \mathit{str}$, $\mathscr{L}(np) = (\mathit{str} \to (\mathit{str}^2 \to \mathit{str}) \to \mathit{str}) \to \mathit{str}$, $\mathscr{L}(s) = \mathit{str}$. The object language $O(\mathscr{G})$ consists of terms representing some English sentences such as *John runs*, *all mice run*, *all cats eat a mouse*, and so on.

14.3 Linearization of Affine ACGs

While linear ACGs can generate languages consisting of linear terms only, affine ACGs can generate languages containing non-linear terms. Therefore, affine ACGs define a strictly richer class of languages than linear ACGs. However, since terms representing strings or trees are linear[3], affine terms in the object languages are not very interesting. This paper shows that for every $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}(m, n)$, we can construct $\mathscr{G}' \in \mathbf{G}^{\mathrm{lin}}(m, \max\{2, n\})$ such that

$$O(\mathscr{G}') = \{\, P \in O(\mathscr{G}) \mid P \text{ is linear} \,\} \tag{14.8}$$

Moreover, in case of $m = 2$, we can find $\mathscr{G}' \in \mathbf{G}^{\mathrm{lin}}(2, n)$ satisfying the equation (14.8). Therefore extending the definition of an ACG to allow lexical entries affine does not enrich the expressive power of ACGs in an essential way. Before proceeding with our construction, we mention a partially stronger result on the special case of this problem on string-generating second-order ACGs, obtained from Salvati's work (Salvati, 2006). He presents an algorithm that converts a linear ACG $\mathscr{G} \in \mathbf{G}^{\mathrm{lin}}(2, n)$ generating a string language into an equivalent LCFRS (via a deterministic tree-walking transducer). Even if an input is an affine ACG $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}(2, n)$, his algorithm still outputs an equivalent LCFRS. Since every LCFRS is encodable by a linear ACG belonging to $\mathbf{G}^{\mathrm{lin}}(2, 4)$ (de Groote and Pogodalla, 2003, 2004), therefore this entails the following corollary.

[3] A string $\mathsf{a}_1 \dots \mathsf{a}_n$ on an alphabet V is represented by $\lambda z^o.\mathsf{a}_1(\dots(\mathsf{a}_n z)\dots) \in \Lambda^{\mathrm{lin}}(\Sigma_V)$ where $\Sigma_V = \langle \{o\}, V, \tau_V \rangle$ with $\tau_V(\mathsf{a}) = \mathit{str}$ for all $\mathsf{a} \in V$ as in Example 1. Trees are constructed on some ranked alphabet. A ranked alphabet $\langle F, \rho \rangle$, where F is an alphabet and ρ is a rank assignment on F, can be identified with a higher-order signature $\Sigma_{\langle F,\rho \rangle} = \langle \{o\}, F, \tau_\rho \rangle$ such that $\tau_\rho(\mathsf{a}) = o^k \to o$ if $\rho(\mathsf{a}) = k$ for all $\mathsf{a} \in F$, and a tree is identified with a variable-free (thus linear) term of the atomic type o in the obvious way.

Corollary 29 *For every string-generating affine ACG $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}(2,n)$, there is a linear ACG $\mathscr{G}' \in \mathbf{G}^{\mathrm{lin}}(2,4)$ such that $O(\mathscr{G}') = O(\mathscr{G})$.*

14.3.1 Basic Idea

We explain our basic idea for the linearization method for affine ACGs through a small example. Let us consider the affine ACG $\mathscr{G}$ consisting of the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}(\mathsf{x})$
A	$p \to s$	$\lambda w^{o^2 \to o}.w\mathsf{a}^o\mathsf{b}^o$
B	p	$\lambda x^o y^o.x$

where $\mathscr{L}(s) = o$ and $\mathscr{L}(p) = o^2 \to o$. Corresponding to $\mathsf{AB} \in \mathscr{A}(\mathscr{G})$, we have $\mathsf{a} \in O(\mathscr{G})$ by

$$\mathscr{L}(\mathsf{AB}) \equiv (\lambda w^{o\to o\to o}.w\mathsf{a}^o\mathsf{b}^o)(\lambda x^o y^o.x) \to_\beta (\lambda x^o y^o.x)\mathsf{a}^o\mathsf{b}^o \twoheadrightarrow_\beta \mathsf{a}^o. \tag{14.9}$$

The occurrences of vacuous λ-abstraction λy^o causes the deletion of b in (14.9). Such deleting operation is what we want to eliminate in order to linearize the affine ACG $\mathscr{G}$. Let us retype λy^o with $\lambda y^{\overline{o}}$ and replace b^o with $\overline{\mathsf{b}}^{\overline{o}}$ to indicate that they should be eliminated. Then (14.9) is decorated by bars as

$$(\lambda w^{o\to\overline{o}\to o}.w\mathsf{a}^o\overline{\mathsf{b}}^{\overline{o}})(\lambda x^o y^{\overline{o}}.x) \to_\beta (\lambda x^o y^{\overline{o}}.x)\mathsf{a}^o\overline{\mathsf{b}}^{\overline{o}} \twoheadrightarrow_\beta \mathsf{a}^o, \tag{14.10}$$

where we retype $w^{o\to o\to o}$ with $w^{o\to\overline{o}\to o}$, so that the whole term is well-typed. In our setting, when a term has a barred type, it means that the term should be erased during β-reduction steps, and vice versa. By eliminating those barred terms and types from (14.10), we get

$$(\lambda w^{o\to o}.w\mathsf{a}^o)(\lambda x^o.x) \to_\beta (\lambda x^o.x)\mathsf{a}^o \to_\beta \mathsf{a}^o, \tag{14.11}$$

which solely consists of linear terms. Hence, the linear ACG $\mathscr{G}'$ with the following lexical entries generates the same language as the original ACG $\mathscr{G}$.

$\mathsf{x} \in \mathscr{C}'_0$	$\tau'_0(\mathsf{x})$	$\mathscr{L}'(\mathsf{x})$
A'	$[p, o \to \overline{o} \to o] \to [s, o]$	$\lambda w^{o\to o}.w\mathsf{a}^o$
B'	$[p, o \to \overline{o} \to o]$	$\lambda x^o.x$

where $[p, o \to \overline{o} \to o]$ and $[s, o]$ are new atomic types that are mapped to $o \to o$ and o, respectively, and $[s, o]$ is the distinguished type. We have $\mathscr{L}(\mathsf{AB}) = \mathscr{L}'(\mathsf{A}'\mathsf{B}')$. The term $\lambda w^{o\to\overline{o}\to o}.w\mathsf{a}^o\overline{\mathsf{b}}^{\overline{o}}$, which is led to $\mathscr{L}'(\mathsf{A}')$, is just one possible bar-decoration for $\mathscr{L}(\mathsf{A})$. For instance, $\lambda w^{\overline{o}\to o\to o}.w\overline{\mathsf{a}}^{\overline{o}}\mathsf{b}^o$ and $\lambda w^{o\to o\to o}.w\mathsf{a}^o\mathsf{b}^o$ are also possible. Bars appearing in $\lambda w^{\overline{o}\to o\to o}.w\overline{\mathsf{a}}^{\overline{o}}\mathsf{b}^o$ predict that the sub-term $\overline{\mathsf{a}}$ will be erased, and $\lambda w^{o\to o\to o}.w\mathsf{a}^o\mathsf{b}^o$ predicts that no subterm of it will disappear. Our linearization method also produces lexical entries corresponding to those bar-decorations.

14.3.2 Formal Definition

We first give a formal definition of the set of possible bar-decorations on a type and a term. Hereafter, we fix a given affine ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$. Define $\overline{\Sigma_1} = \langle \overline{\mathscr{A}_1}, \overline{\mathscr{C}_1}, \overline{\tau_1} \rangle$ by

$$\overline{\mathscr{A}_1} = \{\, \overline{p} \mid p \in \mathscr{A}_1 \,\},\ \overline{\mathscr{C}_1} = \{\, \overline{\mathsf{c}} \mid \mathsf{c} \in \mathscr{C}_1 \,\},\ \overline{\tau_1} = \{\, \overline{\mathsf{c}} \mapsto \overline{\tau_1(\mathsf{c})} \mid \mathsf{c} \in \mathscr{C}_1 \,\},$$

where $\overline{\alpha \to \beta} = \overline{\alpha} \to \overline{\beta}$. Let $\Sigma_1' = \langle \mathscr{A}_1', \mathscr{C}_1', \tau_1' \rangle = \langle \mathscr{A}_1 \cup \overline{\mathscr{A}_1}, \mathscr{C}_1 \cup \overline{\mathscr{C}_1}, \tau_1 \cup \overline{\tau_1} \rangle$. Here, we have the simple lexicon $\widetilde{\cdot}$ from Σ_1' to Σ_1 defined as

$$\widetilde{\overline{p}} = \widetilde{p} = p \text{ for } p \in \mathscr{A}_1, \text{ and } \widetilde{\overline{\mathsf{c}}} \equiv \widetilde{\mathsf{c}} \equiv \mathsf{c} \text{ for } \mathsf{c} \in \mathscr{C}_1.$$

The set $\widehat{\mathcal{T}}(\mathscr{A}_1)$ of possible bar-decorations on types is defined by

$$\widehat{\mathcal{T}}(\mathscr{A}_1) = \{\, \alpha \in \mathcal{T}(\mathscr{A}_1') \mid \text{if } \beta_1 \to \cdots \to \beta_n \to \overline{p} \text{ is a subtype of } \alpha \text{ for some } \overline{p} \in \overline{\mathscr{A}_1}, \text{ then } \beta_1, \ldots, \beta_n \in \mathcal{T}(\overline{\Sigma_1}) \,\}$$

Actually, terms in $\Lambda^{\mathrm{aff}}(\Sigma_1')$ that we are concerned with have types in $\widehat{\mathcal{T}}(\mathscr{A}_1)$. The reason why we ignore types in $\mathcal{T}(\mathscr{A}_1') - \widehat{\mathcal{T}}(\mathscr{A}_1)$ is that if a term is bound to be erased, then so is every sub-term of it. For instance, if a variable x has type $o \to \overline{o} \notin \widehat{\mathcal{T}}(\{o\})$, then the term $x^{o \to \overline{o}} y^o$ has type $\overline{o}$, which, in our setting, means that it should disappear. But if $x^{o \to \overline{o}} y^o$ disappears, so does y^o, which, therefore, should have type $\overline{o}$ to be consistent with our definition.

The set $\widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$ of possible bar-decorations on terms is the subset of $\Lambda^{\mathrm{aff}}(\Sigma_1')$ such that $Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$ iff

- every variable appearing in Q has a type in $\widehat{\mathcal{T}}(\mathscr{A}_1)$, and
- if $\lambda x^\alpha.Q'$ is a sub-term of Q and x^α does not occur free in Q', then $\alpha \in \mathcal{T}(\overline{\mathscr{A}_1})$.

We are not concerned with terms in $\Lambda^{\mathrm{aff}}(\Sigma_1') - \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$.

The following properties are easily seen:

- If $Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$, then $\tau_1'(Q) \in \widehat{\mathcal{T}}(\mathscr{A}_1)$,
- If $\tau_1'(Q) \in \mathcal{T}(\overline{\mathscr{A}_1})$ for $Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$, every sub-term of Q is in $\Lambda^{\mathrm{aff}}(\overline{\Sigma_1})$,
- If $Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$ and $Q \twoheadrightarrow_\beta Q'$, then $Q' \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$.

For each $\alpha \in \mathcal{T}(\mathscr{A}_1)$ and $P \in \Lambda^{\mathrm{aff}}(\Sigma_1)$, Φ gives the set of possible bar-decorations on them:

$$\Phi(\alpha) = \{\, \beta \in \widehat{\mathcal{T}}(\mathscr{A}_1) \mid \widetilde{\beta} = \alpha \,\},$$
$$\Phi(P) = \{\, Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1) \mid \widetilde{Q} \equiv P \,\}.$$

In other words, Φ and $\widetilde{\cdot}$ are inverse of each other, if we disregard types in $\mathcal{T}(\mathscr{A}_1') - \widehat{\mathcal{T}}(\mathscr{A}_1)$ and terms in $\Lambda^{\mathrm{aff}}(\Sigma_1') - \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$.

Secondly, we eliminate barred subtypes from $\alpha \in \widehat{\mathcal{T}}(\mathscr{A}_1) - \mathcal{T}(\overline{\mathscr{A}_1})$ and barred sub-terms from $Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_1})$. Let us define $(\alpha)^\dagger$ and $(Q)^\dagger$ as follows:

$$
\begin{aligned}
(p)^\dagger &= p \quad \text{for } p \in \mathscr{A}_1,\\
(\alpha \to \beta)^\dagger &= \begin{cases} (\alpha)^\dagger \to (\beta)^\dagger & \text{if } \alpha \notin \mathcal{T}(\overline{\mathscr{A}_1}),\\ (\beta)^\dagger & \text{if } \alpha \in \mathcal{T}(\overline{\mathscr{A}_1}),\end{cases}\\
(x^\alpha)^\dagger &\equiv x^{(\alpha)^\dagger},\\
(\mathsf{c})^\dagger &\equiv \mathsf{c} \quad \text{for } \mathsf{c} \in \mathscr{C}_1,\\
(\lambda x^\alpha.Q)^\dagger &\equiv \begin{cases} \lambda x^{(\alpha)^\dagger}.(Q)^\dagger & \text{if } \alpha \notin \mathcal{T}(\overline{\mathscr{A}_1}),\\ (Q)^\dagger & \text{if } \alpha \in \mathcal{T}(\overline{\mathscr{A}_1}),\end{cases}\\
(Q_1Q_2)^\dagger &\equiv \begin{cases} (Q_1)^\dagger(Q_2)^\dagger & \text{if } \tau_1'(Q_2) \notin \mathcal{T}(\overline{\mathscr{A}_1}),\\ (Q_1)^\dagger & \text{if } \tau_1'(Q_2) \in \mathcal{T}(\overline{\mathscr{A}_1}).\end{cases}
\end{aligned}
$$

The following properties are easily seen ($\alpha \in \widehat{\mathcal{T}}(\mathscr{A}_1) - \mathcal{T}(\overline{\mathscr{A}_1})$ and $Q, Q' \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_1})$):

- $(\alpha)^\dagger \in \mathcal{T}(\mathscr{A}_1)$ and $(Q)^\dagger \in \Lambda^{\mathrm{lin}}(\Sigma_1)$,
- $\tau_1((Q)^\dagger) = (\tau_1'(Q))^\dagger$,
- If Q is β-normal, then so is $(Q)^\dagger$,
- $Q =_\beta Q'$ implies $(Q)^\dagger =_\beta (Q')^\dagger$.

Lemma 30 *For every closed term $Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$, $\tau_1'(Q) \in \mathcal{T}(\mathscr{A}_1)$ iff $(Q)^\dagger =_\beta Q =_\beta \widetilde{Q}$.*

Lemma 31 *For every closed term $P \in \Lambda^{\mathrm{aff}}(\Sigma_1)$, $|P|_\beta$ is linear iff there is $Q \in \Phi(P)$ whose type is in $\mathcal{T}(\mathscr{A}_1)$.*

Second-Order Case

We say that an abstract atomic type $p \in \mathscr{A}_0$ is *useless* if there is no $M \in \mathcal{A}(\mathscr{G})$ that has a sub-term whose type contains p. An abstract constant $\mathsf{a} \in \mathscr{C}_0$ is *useless* if there is no $M \in \mathcal{A}(\mathscr{G})$ containing a. If an ACG is second-order, it is easy to check whether the abstract vocabulary contains useless atomic types or constants, and if so, we can eliminate useless abstract atomic types and constants. This can be done in a way similar to the elimination of useless nonterminal symbols and production rules from a context-free grammar.

Definition 15 Let $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s\rangle$ be a second-order ACG that has no useless abstract atomic types or constants. We define $\mathscr{G}' = \langle \Sigma_0', \Sigma_1, \mathscr{L}', [s, \mathscr{L}(s)]\rangle$

as follows: define $\Sigma'_0 = \langle \mathscr{A}'_0, \mathscr{C}'_0, \tau'_0 \rangle$ by

$$\mathscr{A}'_0 = \{\, [p,\beta] \mid p \in \mathscr{A}_0,\ \beta \in \Phi(\mathscr{L}(p)) - \mathcal{T}(\overline{\mathscr{A}_1}) \,\},$$
$$\mathscr{C}'_0 = \{\, [\![\mathsf{a}, Q]\!] \mid \mathsf{a} \in \mathscr{C}_0,\ Q \in \Phi(\mathscr{L}(\mathsf{a})) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_1}) \,\},$$
$$\tau'_0 = \{\, [\![\mathsf{a}, Q]\!] \mapsto ([\tau_0(\mathsf{a}), \tau'_1(Q)])^{\ddagger} \,\},$$
where $([p,\beta])^{\ddagger} = [p,\beta]$,
$$([\alpha \to \gamma, \beta \to \delta])^{\ddagger} = \begin{cases} ([\alpha,\beta])^{\ddagger} \to ([\gamma,\delta])^{\ddagger} & \text{if } \beta \notin \mathcal{T}(\overline{\mathscr{A}_1}), \\ ([\gamma,\delta])^{\ddagger} & \text{if } \beta \in \mathcal{T}(\overline{\mathscr{A}_1}), \end{cases}$$

and $\mathscr{L}'$ by

$$\mathscr{L}'([p,\beta]) = (\beta)^{\dagger}, \qquad \mathscr{L}'([\![\mathsf{a}, Q]\!]) = (Q)^{\dagger}.$$

$\mathscr{G}'$ is linear, but it may contain useless abstract atomic types or constants. The *linearized ACG* $\mathscr{G}^l$ for $\mathscr{G}$ is the result of eliminating all the useless abstract atomic types and constants from $\mathscr{G}'$.

Lemma 32 *Let $\mathscr{G}$ and $\mathscr{G}'$ be as in Definition 15.*

For every variable-free $M \in \Lambda^{\mathrm{lin}}(\Sigma_0)$ of an atomic type and every $Q \in \Phi(\mathscr{L}(M)) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_1})$, there is $N \in \Lambda^{\mathrm{lin}}(\Sigma'_0)$ such that $\tau'_0(N) = [\tau_0(M), \tau'_1(Q)]$ and $\mathscr{L}'(N) \equiv (Q)^{\dagger}$.

Conversely, for every variable-free $N \in \Lambda^{\mathrm{lin}}(\Sigma'_0)$ of an atomic type, there are $M \in \Lambda^{\mathrm{lin}}(\Sigma_0)$ and $Q \in \Phi(\mathscr{L}(M)) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_1})$ such that $\tau'_0(N) = [\tau_0(M), \tau'_1(Q)]$ and $\mathscr{L}'(N) \equiv (Q)^{\dagger}$.

Theorem 33 *For every affine ACG $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}(2,n)$, there is a linear ACG $\mathscr{G}^l \in \mathbf{G}^{\mathrm{lin}}(2,n)$ such that $O(\mathscr{G}^l) = \{\, P \in O(\mathscr{G}) \mid P \text{ is linear} \,\}$.*

Proof. Use Lemmas 31, 32, and 30. □

De Groote and Pogodalla (2003, 2004) have presented encoding methods for linear CFTGs and LCFRSs by linear ACGs. Their methods can also be applied to non-duplicating CFTGs and MCFGs.

Example 2 Let a non-duplicating CFTG G consist of the following productions:[4]

$$S \to P(\mathsf{a},\mathsf{b}), \quad P(x_1,x_2) \to P(\mathsf{c}(x_1),\mathsf{c}(S)) \mid \mathsf{d}(x_1,x_2),$$

where the ranks of S, P, a, b, c, d are 0, 2, 0, 0, 1, 2, respectively. De Groote and Pogodalla's method transforms G into the following affine ACG $\mathscr{G}$:

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}(\mathsf{x})$
A	$p \to s$	$\lambda y_p^{o^2 \to o}.y_p \mathsf{a}^o \mathsf{b}^o$
B	$s \to p \to p$	$\lambda y_s^o y_p^{o^2 \to o} x_1^o x_2^o.y_p(\mathsf{c}^{o \to o} x_1)(\mathsf{c}^{o \to o} y_s)$
C	p	$\lambda x_1^o x_2^o.\mathsf{d}^{o^2 \to o} x_1 x_2$

[4] The notation adopted here follows de Groote and Pogodalla.

When we apply the linearization method given in Definition 15 to $\mathscr{G}$, we get the following linear ACG $\mathscr{G}^l$ whose distinguished type is $[s, o]$:

$\mathsf{x} \in \mathscr{C}_0^l$ $\tau_0^l(\mathsf{x})$	$\mathscr{L}^l(\mathsf{x})$
$[\![\mathsf{A}, \lambda y_p^{o\to o\to o}.y_p\mathsf{ab}]\!]$ $[p, o\to o\to o]\to[s, o]$	$\lambda y_p^{o^2\to o}.y_p\mathsf{ab}$
$[\![\mathsf{A}, \lambda y_p^{o\to\overline{o}\to o}.y_p\mathsf{a}\overline{\mathsf{b}}]\!]$ $[p, o\to\overline{o}\to o]\to[s, o]$	$\lambda y_p^{o\to o}.y_p\mathsf{a}$
$[\![\mathsf{B}, \lambda y_s^o y_p^{o\to o\to o}x_1^o x_2^{\overline{o}}.y_p(\mathsf{c}x_1)(\mathsf{c}y_s)]\!]$ $[s, o]\to[p, o\to o\to o]\to[p, o\to\overline{o}\to o]$	$\lambda y_s^o y_p^{o^2\to o}x_1^o.y_p(\mathsf{c}x_1)(\mathsf{c}y_s)$
$[\![\mathsf{B}, \lambda y_s^{\overline{o}} y_p^{o\to\overline{o}\to o}x_1^o x_2^{\overline{o}}.y_p(\mathsf{c}x_1)(\overline{\mathsf{c}}y_s)]\!]$ $[p, o\to\overline{o}\to o]\to[p, o\to\overline{o}\to o]$	$\lambda y_p^{o\to o}x_1^o.y_p(\mathsf{c}x_1)$
$[\![\mathsf{C}, \lambda x_1^o x_2^o.\mathsf{d}x_1x_2]\!]$ $[p, o\to o\to o]$	$\lambda x_1^o x_2^o.\mathsf{d}x_1x_2$

The linearized ACG $\mathscr{G}^l$ is actually the encoding of the linear CFTG G' consisting of the following productions:

$$S \to P(\mathsf{a}, \mathsf{b}) \mid P'(\mathsf{a}), \quad P'(x_1) \to P(\mathsf{c}(x_1), \mathsf{c}(S)) \mid P'(\mathsf{c}(x_1)),$$
$$P(x_1, x_2) \to \mathsf{d}(x_1, x_2),$$

where the ranks of nonterminals S, P, P' are 0, 2, 1, respectively. G, $\mathscr{G}$, $\mathscr{G}^l$, and G' generate the same tree language.

The following corollary generalizes the result by Fujiyoshi (2005), which covers the *monadic* case only.

Corollary 34 *For every non-duplicating CFTG G, there is a linear CFTG G' such that G and G' generate the same tree language.*

Let $\mathscr{G}$ be the affine ACG that encodes an MCFG G. The linearized ACG $\mathscr{G}^l$ is indeed in the form that is the encoding of an LCFRS[5] (but $\mathscr{G}'$ is not). Therefore, our result covers the following theorem shown by Seki et al. (1991).

Corollary 35 *For every MCFG G, there is an LCFRS G' such that the languages generated by G and G' coincide.*

Third or Higher-Order Case

Definition 15 itself does not depend on the order of the given affine ACG except that in the general case, we do not know how to find and eliminate useless abstract atomic types and constants. For the general case, however, the linearized ACG given in Definition 15 may generate a strictly larger language

[5]The LCFRS obtained from an MCFG through our linearization method may have nonterminals of rank 0. The reason why usual definitions of an LCFRS do not allow nonterminals to have rank 0 is just to avoid redundancy. Mathematically speaking, allowing or disallowing nonterminals of rank 0 does not matter at all.

than the original affine ACG. In the remainder of this paper, we present a linearization method for general affine ACGs.

Example 3 Suppose that an affine ACG $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}(3,1)$ consists of the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}(\mathsf{x})$
A	q	#
B	$p \to q \to q$	$\lambda y^o z^o.\mathsf{b}^{o\to o} z$
C	$q \to s$	$\lambda z^o.z$
D	$(p \to s) \to s$	$\lambda x^{o\to o}.\mathsf{a}^{o\to o}(x\mathsf{e}^o)$

We see $O(\mathscr{G}) = \{\overbrace{\mathsf{a}(\ldots(\mathsf{a}}^{n\text{-times}}\overbrace{(\mathsf{b}(\ldots(\mathsf{b}}^{n\text{-times}}\#)\ldots))\ldots) \mid n \geq 0\}$. The linear ACG $\mathscr{G}'$ by Definition 15 consists of the following lexical entries:

$\mathsf{x} \in \mathscr{C}'_0$	$\tau'_0(\mathsf{x})$	$\mathscr{L}'(\mathsf{x})$
$[\![\mathsf{A}, \#]\!]$	$[q,o]$	#
$[\![\mathsf{B}, \lambda y^{\overline{o}} z^o.\mathsf{b}z]\!]$	$[q,o] \to [q,o]$	$\lambda z^o.\mathsf{b}z$
$[\![\mathsf{C}, \lambda z^o.z]\!]$	$[q,o] \to [s,o]$	$\lambda z^o.z$
$[\![\mathsf{D}, \lambda x^{\overline{o}\to o}.\mathsf{a}(x\overline{\mathsf{e}})]\!]$	$[s,o] \to [s,o]$	$\lambda x^o.\mathsf{a}x$
$[\![\mathsf{D}, \lambda x^{o\to o}.\mathsf{a}(x\mathsf{e})]\!]$	$([p,o] \to [s,o]) \to [s,o]$	$\lambda x^{o\to o}.\mathsf{a}(x\mathsf{e})$

The last lexical entry is useless. We have

$$O(\mathscr{G}') = \{\overbrace{\mathsf{a}(\ldots(\mathsf{a}}^{m\text{-times}}\overbrace{(\mathsf{b}(\ldots(\mathsf{b}}^{n\text{-times}}\#)\ldots))\ldots) \mid m,n \geq 0\} \supsetneq O(\mathscr{G}).$$

Though any term of type p that is the first argument of an occurrence of B is bound to be erased in the original ACG $\mathscr{G}$, we cannot ignore the occurrence of the type p, because that occurrence of p balances the numbers of occurrences of B and D in a term in $\mathscr{A}(\mathscr{G})$.

Our new linearization method gives the linear ACG $\mathscr{G}''$ consisting of the following lexical entries (useless lexical entries are suppressed):

$\mathsf{x} \in \mathscr{C}''_0$	$\tau''_0(\mathsf{x})$	$\mathscr{L}''(\mathsf{x})$
$[\![\mathsf{A}, \#]\!]$	$[q,o]$	#
$[\![\mathsf{B}, \lambda y^{\overline{o}} z^o.\mathsf{b}z]\!]$	$[p,\overline{o}] \to [q,o] \to [q,o]$	$\lambda y^{o\to o} z^o.y(\mathsf{b}z)$
$[\![\mathsf{C}, \lambda z^o.z]\!]$	$[q,o] \to [s,o]$	$\lambda z^o.z$
$[\![\mathsf{D}, \lambda x^{\overline{o}\to o}.\mathsf{a}(x\overline{\mathsf{e}})]\!]$	$([p,\overline{o}] \to [s,o]) \to [s,o]$	$\lambda x^{(o\to o)\to o}.\mathsf{a}(x(\lambda z^o.z))$

where $[p,\overline{o}]$ is mapped to $o \to o$. We have $O(\mathscr{G}) = O(\mathscr{G}'')$.

Now, we give the formal definition of our new linearization method for general affine ACGs. For simplicity, we assume that $\mathscr{A}_1 = \{o\}$ here, but it is possible to lift this assumption. The new linearized ACG $\mathscr{G}''$ has the form

$\mathscr{G}'' = \langle \Sigma_0'', \Sigma_1, \mathscr{L}'', [s, \mathscr{L}(s)]\rangle$, where $\Sigma_0'' = \langle \mathscr{A}_0'', \mathscr{C}_0'', \tau_0''\rangle$ is defined by

$$\begin{aligned}
\mathscr{A}_0'' &= \{\, [p,\beta] \mid p \in \mathscr{A}_0,\ \beta \in \Phi(\mathscr{L}(p)) \,\},\\
\mathscr{C}_0'' &= \{\, [\![\mathsf{a}, Q]\!] \mid \mathsf{a} \in \mathscr{C}_0,\ Q \in \Phi(\mathscr{L}(\mathsf{a})) \,\},\\
\tau_0'' &= \{\, [\![\mathsf{a}, Q]\!] \mapsto [\tau_0(\mathsf{a}), \tau_1'(Q)] \,\}\\
&\quad \text{where } [\alpha \to \gamma, \beta \to \delta] = [\alpha,\beta] \to [\gamma,\delta].
\end{aligned}$$

Here we have two simple lexicons $\mathscr{L}_0 : \Sigma_0'' \to \Sigma_0$ and $\mathscr{L}_1 : \Sigma_0'' \to \Sigma_1'$;

$$\mathscr{L}_0([p,\beta]) = p, \quad \mathscr{L}_0([\![\mathsf{a}, Q]\!]) = \mathsf{a}, \quad \mathscr{L}_1([p,\beta]) = \beta, \quad \mathscr{L}_1([\![\mathsf{a}, Q]\!]) = Q.$$

We have $\widetilde{\mathscr{L}_1(N)} \equiv \mathscr{L} \circ \mathscr{L}_0(N)$ for $N \in \Lambda^{\mathrm{lin}}(\Sigma_0'')$. For any $M \in \Lambda^{\mathrm{lin}}(\Sigma_0)$ and $Q \in \Phi(\mathscr{L}(M))$, one can find a term $\chi(M,Q) \in \Lambda^{\mathrm{lin}}(\Sigma_0'')$ such that $\mathscr{L}_0(\chi(M,Q)) \equiv M$ and $\mathscr{L}_1(\chi(M,Q)) \equiv Q$.

Lemma 36 *For every $Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$ and $\alpha \in \mathcal{T}(\mathscr{A}_0)$, the following statements are equivalent:*

1. *There is $M \in \Lambda^{\mathrm{lin}}(\Sigma_0)$ of type α such that $\mathscr{L}(M) \equiv \widetilde{Q}$.*
2. *There is $N \in \Lambda^{\mathrm{lin}}(\Sigma_0'')$ of type $[\alpha, \tau_1'(Q)]$ such that $\mathscr{L}_1(N) \equiv Q$.*

Lemmas 31 and 36 imply

$$\{\, M \in \mathcal{A}(\mathscr{G}) \mid |\mathscr{L}(M)|_\beta \text{ is linear} \,\} = \{\, \mathscr{L}_0(N) \mid N \in \mathcal{A}(\mathscr{G}'') \,\}.$$

Since $(\mathscr{L}_1(N))^\dagger =_\beta \widetilde{\mathscr{L}_1(N)} \equiv \mathscr{L} \circ \mathscr{L}_0(N)$ for every $N \in \mathcal{A}(\mathscr{G}'')$ by Lemma 30, it is enough to define a new lexicon $\mathscr{L}''$ so that

$$\mathscr{L}''(N) =_{\beta\eta} (\mathscr{L}_1(N))^\dagger \tag{14.12}$$

for every $N \in \mathcal{A}(\mathscr{G}'')$.

We define the type substitution $\sigma : \mathscr{A}_0'' \to \mathcal{T}(\{o\})$ of $\mathscr{L}'' = \langle \sigma, \theta\rangle$ as

$$\sigma([p,\beta]) = \begin{cases} (\beta)^\dagger & \text{if } \beta \notin \mathcal{T}(\{\overline{o}\}),\\ o \to o & \text{if } \beta \in \mathcal{T}(\{\overline{o}\}). \end{cases}$$

Here we identify σ with its homomorphic extension. As a preparation for defining the term substitution θ of $\mathscr{L}''$, we give three kinds of linear combinators. For $[\alpha,\beta] \in \mathcal{T}(\mathscr{A}_0'')$ such that $\beta \in \mathcal{T}(\{\overline{o}\})$, let $\sigma([\alpha,\beta]) = \gamma_1 \to \cdots \to \gamma_m \to o \to o$ and $\gamma_i = \gamma_{i,1} \to \cdots \to \gamma_{i,k_i} \to o \to o$. $Z^{\sigma([\alpha,\beta])}$ is a linear combinator of type $\sigma([\alpha,\beta])$ defined as

$$\begin{aligned}
Z^{\sigma([\alpha,\beta])} &\equiv \lambda y_1^{\gamma_1} \ldots y_m^{\gamma_m} z^o.R_1(R_2(\ldots(R_m z)\ldots))\\
&\quad \text{where } R_i \equiv y_i^{\gamma_i} Z^{\gamma_{i,1}} \ldots Z^{\gamma_{i,k_i}}.
\end{aligned}$$

For each $[\alpha,\beta] \in \mathcal{T}(\mathscr{A}_0'')$ such that $\beta \in \widehat{\mathcal{T}}(\{o\}) - \mathcal{T}(\{\overline{o}\})$, we define two linear combinators X_α^β of type $\sigma([\alpha,\beta]) \to (\beta)^\dagger$ and Y_α^β of type $(\beta)^\dagger \to \sigma([\alpha,\beta])$ by mutual induction. Let $[\alpha,\beta] = [\alpha_1,\beta_1] \to \cdots \to [\alpha_m,\beta_m] \to [p,\beta_0]$ with $[p,\beta_0] \in \mathscr{A}_0''$ and the set $\{1,\ldots,m\}$ be partitioned into two subsets I and J so

that $\beta_i \notin \mathcal{T}(\{\overline{o}\})$ iff $i \in I$. Let $I = \{i_1, \ldots, i_k\}$ $(i_j < i_{j+1})$ and $J = \{j_1, \ldots, j_l\}$. Let

$$X^{\beta}_{\alpha} \equiv \lambda y^{\sigma([\alpha,\beta])} x_{i_1}^{(\beta_{i_1})^{\dagger}} \ldots x_{i_k}^{(\beta_{i_k})^{\dagger}} . y^{\sigma([\alpha,\beta])} P_1 \ldots P_m$$

$$\text{where} \quad P_i \equiv \begin{cases} Y^{\beta_i}_{\alpha_i} x_i^{(\beta_i)^{\dagger}} & \text{if } i \in I, \\ Z^{\sigma([\alpha_i,\beta_i])} & \text{if } i \in J, \end{cases}$$

and

$$Y^{\beta}_{\alpha} \equiv \lambda x^{(\beta)^{\dagger}} y_1^{\sigma([\alpha_1,\beta_1])} \ldots y_m^{\sigma([\alpha_m,\beta_m])} \vec{z}. M_{j_1}(\ldots(M_{j_l}(x^{(\beta)^{\dagger}} L_{i_1} \ldots L_{i_k} \vec{z}))\ldots)$$

where $\vec{z}$ is short for $z_1^{\gamma_1} \ldots z_n^{\gamma_n}$ for $(\beta_0)^{\dagger} = \gamma_1 \to \cdots \to \gamma_n \to o$, and

$$\begin{cases} L_i \equiv X^{\beta_i}_{\alpha_i} y_i^{\sigma([\alpha_i,\beta_i])} & \text{for } i \in I, \\ M_i \equiv Z^{\sigma([\alpha_i,\beta_i])\to o\to o} y_i^{\sigma([\alpha_i,\beta_i])} & \text{for } i \in J. \end{cases}$$

Note that if $[\alpha,\beta] = [p,\beta_0] \in \mathscr{A}_0''$, then $X^{\beta_0}_{p} =_{\beta\eta} Y^{\beta_0}_{p} =_{\beta\eta} \lambda z^{(\beta_0)^{\dagger}} . z$.

Now, we give a new linearization method as follows.

Definition 16 For a given affine ACG $\mathscr{G}$, we define a new linear ACG as $\mathscr{G}'' = \langle \Sigma_0'', \Sigma_1, \mathscr{L}'', [s, \mathscr{L}(s)]\rangle$, where $\mathscr{L}'' = \langle \sigma, \theta \rangle$ for σ as above and

$$\theta(\llbracket \mathsf{a}, Q \rrbracket) \equiv \begin{cases} |Y^{\tau_1'(Q)}_{\tau_0(\mathsf{a})}(Q)^{\dagger}|_{\beta} & \text{if } \tau_1'(Q) \notin \mathcal{T}(\{\overline{o}\}), \\ Z^{\sigma(\tau_0''(\llbracket \mathsf{a},Q \rrbracket))} & \text{if } \tau_1'(Q) \in \mathcal{T}(\{\overline{o}\}). \end{cases}$$

If $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}(m,n)$, then $\mathscr{G}'' \in \mathbf{G}^{\mathrm{lin}}(m, \max\{2,n\})$.

Lemma 37 *Given $N \in \Lambda(\Sigma_0'')$ of type $[\alpha,\beta]$ such that $\beta \notin \mathcal{T}(\{\overline{o}\})$ and $\mathscr{L}_1(N) \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$, we have*

$$(\mathscr{L}_1(N))^{\dagger} =_{\beta\eta} X^{\beta}_{\alpha} \mathscr{L}''(N) \phi_N$$

where ϕ_N is the substitution on the free variables of $\mathscr{L}''(N)$ such that

$$x^{\sigma([\alpha,\beta])}\phi_N = \begin{cases} Y^{\beta}_{\alpha} x^{(\beta)^{\dagger}} & \textit{if } x \textit{ has the type } [\alpha,\beta] \textit{ in } N \textit{ and } \beta \notin \mathcal{T}(\{\overline{o}\}), \\ Z^{\sigma([\alpha,\beta])} & \textit{otherwise.} \end{cases}$$

Theorem 38 *For every affine ACG $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}(m,n)$, there is a linear ACG $\mathscr{G}'' \in \mathbf{G}^{\mathrm{lin}}(m, \max\{2,n\})$ such that $O(\mathscr{G}'') = \{ P \in O(\mathscr{G}) \mid P \textit{ is linear} \}$.*

Proof. Lemma 37 entails the equation (14.12). □

14.4 Concluding Remarks

We have shown that the generative capacity of linear ACGs is as rich as that of affine ACGs, that is, the non-deletion constraint on linear ACGs is superficial. Our linearization method, however, increases the size of the given grammar exponentially due to the definition of Φ, so there may still exist an advantage of allowing deleting operations in the ACG formalism. For instance, the

atomic type *np* of the abstract vocabulary of the ACG in Example 1 will be divided up into three new atomic types which correspond to noun phrases as third person singular subjects, plural subjects, and objects, respectively.

One attractive feature of ACGs is that they can be thought of as a generalization of several well-established grammar formalisms (de Groote, 2002, de Groote and Pogodalla, 2003, 2004). This paper demonstrates that the ACG formalism also generalizes some "operation" on those grammars, namely, conversion from non-duplicating grammars into non-duplicating and non-deleting ones.

Recall that Fisher (1968a,b) showed that every CFTG has a corresponding non-deleting CFTG whose string IO-language is equivalent. As a generalization of his result, the author conjectures that one can eliminate vacuous λ-abstraction from *semi-affine ACGs* preserving the orders of the abstract vocabularies and the lexicons, where a term is *semi-affine* if for every free variable x of any sub-term, either x occurs at most once, or x has at most second-order type. Actually, every CFTG has a corresponding semi-affine ACG such that the tree IO-language of the CFTG coincides with the object language of the ACG, and the semi-affine ACG encoding a non-deleting CFTG has no vacuous λ-abstraction. If the conjecture is correct, this implies that every CFTG has a corresponding non-deleting CFTG whose tree/string IO-language is equivalent.

Acknowledgment

The author is grateful to Makoto Kanazawa for initiating this research and giving advice throughout this work. The author would like to thank to Sylvain Salvati for his invaluable comments on the draft of this paper. In particular, he inspired the author to get the conjecture stated in the last section.

References

de Groote, Philippe. 2001. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pages 148–155.

de Groote, Philippe. 2002. Tree-adjoining grammars as abstract categorial grammars. In *TAG+6, Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 145–150. Università di Venezia.

de Groote, Philippe and Sylvain Pogodalla. 2003. *m*-linear context-free rewriting systems as abstract categorial grammars. In R. T. Oehrle and J. Rogers, eds., *Proceedings of Mathematics of Language - MOL-8, Bloomington, Indiana, U. S.*, pages 71–80.

de Groote, Philippe and Sylvain Pogodalla. 2004. On the expressive power of abstract categorial grammars: Representing context-free formalisms. *Journal of Logic, Language and Information* 13(4):421–438.

Fisher, Michael J. 1968a. *Grammars with Macro-Like Productions*. Ph.D. thesis, Harvard University.

Fisher, Michael J. 1968b. Grammars with macro-like productions. In *Proceedings of the 9th IEEE Conference on Switching and Automata Theory*, pages 131–142.

Fujiyoshi, Akio. 2005. Linearity and nondeletion on monadic context-free tree grammars. *Information Processing Letters* 93(3):103–107.

Hindley, J. Roger. 1997. *Basic Simple Type Theory*. Cambridge University Press.

Kanazawa, Makoto and Ryo Yoshinaka. 2005. Lexicalization of second-order ACGs. Tech. Rep. NII-2005-012E, National Institute of Informatics.

Salvati, Sylvain. 2006. Encoding second order string ACGs with deterministic tree walking transducers. In *Proceedings of the 11th conference on Formal Grammar*.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88(2):191–229.