# 1

# Generating Control Languages with Abstract Categorial Grammars

Makoto Kanazawa and Sylvain Salvati †

**Abstract**

We show that the class of level-$k$ control languages, as defined by Weir, is properly included in the class of $2^{k-1}$-multiple context-free languages for each $k \geq 2$. The proof of inclusion uses a representation of the set of derivation trees for a level-$k$ control language in terms of a second-order abstract categorial grammar.

**Keywords** control languages, multiple context-free grammars, abstract categorial grammars

## 1.1 Introduction

Weir (1992a) introduced a hierarchy of language classes known as the *control language hierarchy*, starting with the context-free languages at level one. Control languages at level two and above are each generated by a context-free grammar coupled with a language at the previous level, which controls derivation trees of the grammar in a certain "linear" fashion. Thus, for $k \geq 2$, level-$k$ control languages are defined in terms of non-regular sets of derivation trees and the usual simple yield function, which merely concatenates yields of subtrees. Even though it is known that the class of level-2 control languages coincides with the class of *tree-adjoining languages* or *head languages* (Weir 1992a, Vijay-Shanker and Weir 1994), it is not immediately clear how the higher levels are related to language classes generated by grammar formalisms that have regular sets of derivation trees and complex yield

†The second author's work for this paper was done while he was at the National Institute of Informatics as a postdoctoral fellow of the Japan Society for the Promotion of Science, and was supported by the Grant-in-Aid for Scientific Research (18-06739).

*FG-2007*.
Organizing Committee:, Laura Kallmeyer, Paola Monachesi, Gerald Penn, Giorgio Satta.

functions, such as *multiple context-free grammars* (Seki et al. 1991).

In this paper, we prove that the class of level-$k$ control languages is properly included in the class of $2^{k-1}$-multiple context-free languages for each $k \geq 2$. The proof of inclusion uses a representation of the set of derivation trees for a control language in terms of a second-order *abstract categorial grammar* (de Groote 2001). Composing this tree-generating ACG with the yield function gives a string-generating second-order ACG, whose language must be a multiple context-free language, as shown by Salvati (2007) using the equivalence between MCFLs and output languages of *deterministic tree-walking transducers* (Weir 1992b). We here give a direct conversion from a second-order ACG of *width* $2m$ to an $m$-multiple context-free grammar to show that a level-$k$ control language is a $2^{k-1}$-MCFL.

Our motivation for relating control languages and MCFLs comes from pumping lemmas for the two formalisms. For MCFLs, Seki et al. (1991) gave a rather weak pumping lemma, which only says that an infinite $m$-MCFL contains a string that can be pumped at some $2m$ positions. However, Radzinski (1991) and Groenink (1997) erroneously credited Seki et al. (1991) with (implicitly) proving the following much stronger statement:[1]

**Myth 1** *Let $L$ be an $m$-MCFL. There is a constant $n$ such that for every $z \in L$, if $|z| \geq n$, then $z$ may be written as $z = u_1 v_1 w_1 s_1 u_2 v_2 w_2 s_2 \dots u_m v_m w_m s_m u_{m+1}$ in such a way as to satisfy the following conditions:*

(i) $\sum_{j=1}^{m} |v_j s_j| \geq 1$;
(ii) $u_1 v_1^i w_1 s_1^i u_2 v_2^i w_2 s_2^i \dots u_m v_m^i w_m s_m^i u_{m+1} \in L$ *for all* $i \geq 0$.

Contrary to Radzinski's and Groenink's claims, the proof given by Seki et al. (1991) cannot be easily converted to a proof of the above statement. It is an open question whether Myth 1 holds for any $m \geq 2$.

As for the control language hierarchy, Palis and Shende (1995) proved an Ogden-style pumping lemma, which implies the following statement:

**Theorem 2 (Palis and Shende)** *Let $L$ be a level-$k$ control language. There is a constant $n$ such that for every $z \in L$, if $|z| \geq n$, then $z$ may be written as $z = u_1 v_1 w_1 s_1 u_2 v_2 w_2 s_2 \dots u_{2^{k-1}} v_{2^{k-1}} w_{2^{k-1}} s_{2^{k-1}} u_{2^{k-1}+1}$ in such a way as to satisfy the following conditions:*

(i) $\sum_{j=1}^{2^{k-1}} |v_j s_j| \geq 1$;
(ii) $u_1 v_1^i w_1 s_1^i u_2 v_2^i w_2 s_2^i \dots u_{2^{k-1}} v_{2^{k-1}}^i w_{2^{k-1}} s_{2^{k-1}}^i u_{2^{k-1}+1} \in L$ *for all* $i \geq 0$;
(iii) $|s_{2^{k-2}} u_{2^{k-2}+1} v_{2^{k-2}+1}| \leq n$.

Given the form of Myth 1 and of Theorem 2, it is natural to conjecture that the class of level-$k$ control languages is included in the class of $2^{k-1}$-MCFLs.

[1] Radzinski (1991) refers to an earlier technical report (Kasami et al. 1987), which was later incorporated into Seki et al. 1991.

We prove this conjecture, and use Theorem 2 to prove that the inclusion is proper for $k \geq 2$. Note that the special case of this result for $k = 2$ is already known, since Seki et al. (1991) showed that the class of 2-MCFLs properly includes the class of head languages.

## 1.2 Preliminaries

### 1.2.1 Tree languages

A *ranked alphabet* is a finite set $F = \bigcup_{n\in\mathbb{N}} F^{(n)}$, where $F^{(m)} \cap F^{(n)} = \varnothing$ if $m \neq n$. If $f \in F^{(n)}$, $n$ is the *rank* of $f$, written $\mathrm{rank}(f)$. A *tree* over $F$ is an expression $fT_1 \dots T_n$, where $\mathrm{rank}(f) = n$ and $T_1, \dots, T_n$ are trees over $F$. The set of trees over $F$ is denoted $\mathbb{T}_F$. Subsets of $\mathbb{T}_F$ are called *tree languages*.

Let $F$ be a ranked alphabet, $c \in F^{(0)}$, and $L_1, L_2 \subseteq \mathbb{T}_F$. The *concatenation* of $L_1$ to $L_2$ through $c$, written $L_2 \cdot_c L_1$, is $\bigcup_{T\in L_2} T[c \leftarrow L_1]$, where $T[c \leftarrow L]$ is defined inductively as follows:[2]

$$c[c \leftarrow L] = L,$$
$$(fT_1 \dots T_n)[c \leftarrow L] = \{\, fU_1 \dots U_n \mid U_i \in T_i[c \leftarrow L] \text{ for } i = 1, \dots, n \,\} \quad \text{if } f \neq c, \text{ where } n = \mathrm{rank}(f).$$

The *Kleene star* of $L$ (through $c$), written $L^{*,c}$, is defined to be $\bigcup_{n\geq 0} L^{n,c}$, where

$$L^{0,c} = \{c\}, \qquad L^{n+1,c} = L^{n,c} \cup L \cdot_c L^{n,c}.$$

See Gécseg and Steinby 1997 or Comon et al. 2002 for the definitions of *regular tree language* and of *(linear non-deleting) tree homomorphism*. The set of derivation trees of a context-free grammar is a special kind of regular tree language known as a *local set*.

### 1.2.2 Control language hierarchy

A *headed ranked alphabet* is a pair $(F, h)$, where

- $F$ is a ranked alphabet;
- $h$ is a function from $F$ to $\mathbb{N}$ such that $0 \leq h(f) \leq \mathrm{rank}(f)$.

Let $(F, h)$ be a headed ranked alphabet. If $T$ is a tree over $F$, the *spine* of $T$, written $\mathrm{spine}(T)$, is a string in $F^*$ defined as follows:[3]

$$\mathrm{spine}(fT_1 \dots T_n) = \begin{cases} f & \text{if } h(f) = 0, \\ f\ \mathrm{spine}(T_j) & \text{if } h(f) = j \geq 1. \end{cases}$$

The set of all (maximal) spines of subtrees of $T$ is then $\mathrm{Spines}(T) = \{\mathrm{spine}(T)\} \cup \mathrm{iSpines}(T)$, where $\mathrm{iSpines}(T)$, the set of *interior spines* of $T$,

[2] Note that the concatenation of $L_1$ to $L_2$ through $c$ is written $L_1 \cdot_c L_2$ in Gécseg and Steinby 1997. Our notation follows Comon et al. (2002).

[3] $F^*$ is the set of strings over $F$ regarded as an unranked alphabet.

is defined as follows:

$$\mathrm{iSpines}(fT_1 \dots T_n) = \begin{cases} \bigcup_{1 \le i \le n} \mathrm{Spines}(T_i) & \text{if } h(f) = 0, \\ \mathrm{iSpines}(T_j) \cup \bigcup_{1 \le i \le n, i \ne j} \mathrm{Spines}(T_i) & \text{if } h(f) = j \ge 1. \end{cases}$$

A *labeled distinguished grammar* (LDG) over a terminal alphabet $V$ is a six-tuple $G = (N, V, P, S, F, h)$, where $G^\circ = (N, V, P, S)$ is a context-free grammar, $(F, h)$ is a headed ranked alphabet, and

- $F = \{ f_\pi \mid \pi \in P \}$ (i.e., the symbols in $F$ name the rules in $P$);
- $\mathrm{rank}(f_\pi)$ is the number of occurrences of nonterminals on the right-hand side of $\pi$.

The *rule trees* of $G$ are defined inductively as follows:

- If $\pi = B \to w_0 B_1 w_1 \dots B_n w_n$ with $w_0, \dots, w_n \in V^*$ and $B, B_1, \dots, B_n \in N$, then the tree $f_\pi T_1 \dots T_n$ is a rule tree of type $B$ if $T_1, \dots, T_n$ are rule trees of type $B_1, \dots, B_n$, respectively.

A rule tree of type $S$ is a *complete rule tree*. The set of complete rule trees of an LDG is a local set. A rule tree $T$ determines a derivation tree of $G^\circ$, and one can associate with $T$ its *yield*, $\mathrm{yield}_G(T)$, in an obvious way.

The language $L(G, C)$ generated by a labeled distinguished grammar $G = (N, V, P, S, F, h)$ and a *control set* $C \subseteq F^*$ is[4]

$$\{ \mathrm{yield}_G(T) \mid T \text{ is a complete rule tree of } G \text{ and } \mathrm{Spines}(T) \subseteq C \}.$$

The class $\mathbf{C}_k$ of *k-level control languages* (Weir 1992a) is defined as follows:

$$\mathbf{C}_1 = \mathrm{CFL}, \qquad \mathbf{C}_{k+1} = \{ L(G, C) \mid G \text{ is an LDG and } C \in \mathbf{C}_k \}.$$

It is known that for each $k$, $\mathbf{C}_k$ is a full abstract family of languages and all languages in $\mathbf{C}_k$ belong to the complexity class LOGCFL (Weir 1992a, Palis and Shende 1992).

### 1.2.3 Multiple context-free grammars

A *multiple context-free grammar* (Seki et al. 1991) is a tuple $G = (N, V, P, S)$, where $N$ is a ranked alphabet of *nonterminals*, $V$ is an (unranked) alphabet of *terminals*, $S \in N^{(1)}$, and $P$ is a set of *rules* of the form

$$B(t_1, \dots, t_r) :- B_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B_n(x_{n,1}, \dots, x_{n,r_n}),$$

where $n \ge 0$, $B, B_1, \dots, B_n$ are nonterminals of rank $r, r_1, \dots, r_n$, respectively, $x_{i,j}$ are pairwise distinct variables, and $t_1, \dots, t_r$ are strings over $V \cup \{ x_{i,j} \mid 1 \le i \le n, 1 \le j \le r_i \}$ such that each $x_{i,j}$ occurs at most once in $t_1 \dots t_r$. The symbol $:-$ is omitted when $n = 0$. We say that $G$ is an $m$-MCFG if the rank

[4] The present definition of $L(G, C)$ is not exactly equivalent to Weir's (1992a), but it is easy to see that it leads to an equivalent definition of the control language hierarchy.

of nonterminals does not exceed $m$. The language of $G$ is $L(G) = \{ w \in V^* \mid \vdash_G S(w) \}$, where $\vdash_G$ is defined as follows:

- if $B(t_1, \ldots, t_r) :- B_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, B_n(x_{n,1}, \ldots, x_{n,r_n})$ is a rule in $P$ and $\vdash_G B_i(w_{i,1}, \ldots, w_{i,r_i})$ for $i = 1, \ldots, n$, then $\vdash_G B(t_1\sigma, \ldots, t_r\sigma)$, where $\sigma$ is the substitution that sends $x_{i,j}$ to $w_{i,j}$.

**Example 1** The 2-MCFG consisting of the following five rules generates $\text{RESP} = \{ a_1^m a_2^m b_1^n b_2^n a_3^m a_4^m b_3^n b_4^n \mid m, n \geq 0 \}$:

$$S(x_{1,1}x_{2,1}x_{1,2}x_{2,2}) :- P(x_{1,1}, x_{1,2}), Q(x_{2,1}, x_{2,2}). \qquad P(\epsilon, \epsilon). \qquad Q(\epsilon, \epsilon).$$
$$P(a_1 x_{1,1} a_2, a_3 x_{1,2} a_4) :- P(x_{1,1}, x_{1,2}). \quad Q(b_1 x_{1,1} b_2, b_3 x_{1,2} b_4) :- Q(x_{1,1}, x_{1,2}).$$

We know from Seki et al. 1991 that $\text{RESP} \in \text{2-MCFL} - \mathbf{C}_2$.

An MCFG is *non-erasing* if for each rule, every variable on the right-hand side occurs exactly once on the left-hand side. Non-erasing MCFGs are also known as (string-based) *linear context-free rewriting systems* (Vijay-Shanker et al. 1987). Seki et al. (1991) show that every $m$-MCFG has an equivalent non-erasing $m$-MCFG.

### 1.2.4 Types and $\lambda$-terms

Given a finite set $A$ of *atomic types*, we let $\mathscr{T}(A)$ denote the set of *types* built up from atomic types using the type constructor $\to$.[5] The *size* of a type $\alpha$ is the number of atomic type occurrences in it and is written $|\alpha|$. The *order* of a type $\alpha$, denoted by $\text{ord}(\alpha)$, is defined as follows:

$$\text{ord}(p) = 1 \quad \text{if } p \text{ is atomic}, \qquad \text{ord}(\alpha \to \beta) = \max(\text{ord}(\alpha) + 1, \text{ord}(\beta)).$$

A *higher-order signature* is a triple $\Sigma = (A, C, \tau)$, where $A$ is a finite set of atomic types, $C$ is a finite set of *constants*, and $\tau$ is a mapping from $C$ to $\mathscr{T}(A)$. The *order* of $\Sigma$ is $\max\{ \text{ord}(\tau(c)) \mid c \in C \}$. Let $X$ be a countably infinite set of variables. The set $\Lambda(\Sigma)$ of (untyped) *$\lambda$-terms* over a higher-order signature $\Sigma = (A, C, \tau)$ is the smallest superset of $X \cup C$ satisfying the following conditions:[6]

1. If $M, N \in \Lambda(\Sigma)$, then $(MN) \in \Lambda(\Sigma)$;
2. If $M \in \Lambda(\Sigma)$ and $x \in X$, then $(\lambda x.M) \in \Lambda(\Sigma)$.

The set $\text{FV}(M)$ of *free variables* of $M$ is understood in the usual way. A $\lambda$-term $M$ is *closed* if $\text{FV}(M) = \varnothing$; it is *pure* if it contains no constants.

A $\lambda$-term may be assigned a type under a *type environment*, which is a finite set $\Gamma$ of variable declarations of the form $x{:}\alpha$ (where $x \in X, \alpha \in \mathscr{T}(A)$) in

[5] The connective $\to$ is assumed to be right-associative, so we write $\alpha_1 \to \alpha_2 \to \alpha_3$ for $\alpha_1 \to (\alpha_2 \to \alpha_3)$. The notation $\alpha^k \to \beta$ abbreviates $\alpha \to \cdots \to \alpha \to \beta$ with "$\alpha\to$" repeated $k$ times.

[6] As usual, we omit the outermost parentheses and write $MNP$ for $(MN)P$, $\lambda x.MN$ for $\lambda x.(MN)$, and $\lambda x_1 \ldots x_n.M$ for $\lambda x_1.(\lambda x_2.\ldots(\lambda x_n.M)\ldots)$. The notation $MN^k$ abbreviates $MN \ldots N$ with "$N$" repeated $k$ times.

which no variable is declared more than once. A type environment is usually written as a list $x_1 : \alpha_1, \ldots, x_n : \alpha_n$. The following inference system, $\lambda{\to_\Sigma}$, derives *typing judgments* of the form $\Gamma \vdash_\Sigma M : \alpha$, where $\Gamma$ is a type environment, $M \in \Lambda(\Sigma)$, and $\alpha \in \mathscr{T}(A)$:

$$\vdash_\Sigma c : \tau(c) \quad \text{for } c \in C, \qquad x : \alpha \vdash_\Sigma x : \alpha \quad \text{for } x \in X \text{ and } \alpha \in \mathscr{T}(A),$$

$$\frac{\Gamma \vdash_\Sigma M : \beta}{\Gamma - \{x : \alpha\} \vdash_\Sigma \lambda x.M : \alpha \to \beta} \quad \text{if } \Gamma \cup \{x : \alpha\} \text{ is a type environment,}$$

$$\frac{\Gamma \vdash_\Sigma M : \alpha \to \beta \quad \Delta \vdash_\Sigma N : \alpha}{\Gamma \cup \Delta \vdash_\Sigma MN : \beta} \quad \text{if } \Gamma \cup \Delta \text{ is a type environment.}$$

We write $\Gamma \vdash M : \alpha$ when $M$ is pure, omitting reference to $\Sigma$.

A $\lambda$-term $M$ is *linear* if the following conditions both hold:

1. for any subterm $\lambda x.N$ of $M$, $x \in \mathrm{FV}(N)$;
2. for any subterm $NP$ of $M$, $\mathrm{FV}(N) \cap \mathrm{FV}(P) = \varnothing$.

We denote the set of linear $\lambda$-terms over $\Sigma$ by $\Lambda_{\mathrm{lin}}(\Sigma)$. For every $\Gamma$ and $\alpha$, there are only finitely many pure linear $\lambda$-terms $M$ in $\beta$-normal form such that $\Gamma \vdash M : \alpha$.

See Hindley 1997 or Sørensen and Urzyczyn 2006 for other standard notions in simply-typed $\lambda$-calculus, such as *substitution* (of a $\lambda$-term for a free variable in a $\lambda$-term), *$\beta$-reduction*, and *$\beta$-normal form*. We write $\twoheadrightarrow_\beta$ for $\beta$-reduction, and $=_\beta$ for $\beta$-equality. We denote the $\beta$-normal form of $M$ by $|M|_\beta$.

### 1.2.5 Strings and trees as linear $\lambda$-terms

A string $a_1 \ldots a_n$ over an (unranked) alphabet $V$ can be represented by a closed $\lambda$-term $/a_1 \ldots a_n/ = \lambda z.a_1(\ldots(a_n z)\ldots)$ in $\Lambda_{\mathrm{lin}}(\Sigma_V^{\mathrm{string}})$, where $\Sigma_V^{\mathrm{string}} = (\{o\}, V, \tau)$ and $\tau(a) = o \to o$ for all $a \in V$. We call $\Sigma_V^{\mathrm{string}}$ a *string signature*. Note that $\vdash_{\Sigma_V^{\mathrm{string}}} /w/ : o \to o$ for all strings $w \in V^*$. Concatenation of strings can be represented by the combinator $\mathbf{B} = \lambda xyz.x(yz)$. We have $\mathbf{B}\, M(\mathbf{B}\, NP) =_\beta \mathbf{B}\,(\mathbf{B}\, MN)P$, and for strings $v, w \in V^*$, $\mathbf{B}\, /v/\,/w/ \twoheadrightarrow_\beta /vw/$.

A ranked alphabet $F$ can be represented by a second-order signature $\Sigma_F^{\mathrm{tree}} = (\{o\}, F, \tau_F)$, where for each $f \in F^{(n)}$, $\tau_F(f) = o^n \to o$. We call $\Sigma_F^{\mathrm{tree}}$ a *tree signature*. We identify a tree in $\mathbb{T}_F$ with a closed $\beta$-normal $\lambda$-term in $\Lambda_{\mathrm{lin}}(\Sigma_F^{\mathrm{tree}})$ of type $o$ in the obvious way.

An alternative representation of a string $a_1 \ldots a_n$ is by means of a *monadic tree* $\ulcorner a_1 \ldots a_n \urcorner = a_1^{(1)}(\ldots(a_{n-1}^{(1)} a_n^{(0)})\ldots)$, which is a linear $\lambda$-term over $\Sigma_F^{\mathrm{tree}}$, where $F = F^{(0)} \cup F^{(1)} = \{\, a^{(0)} \mid a \in V \,\} \cup \{\, a^{(1)} \mid a \in V \,\}$.

### 1.2.6 Abstract categorial grammars

When we write $\Sigma, \Sigma', \Sigma_1$, etc., to refer to higher-order signatures, we assume $\Sigma = (A, C, \tau)$, $\Sigma' = (A', C', \tau')$, $\Sigma_1 = (A_1, C_1, \tau_1)$, etc. Given higher-order

signatures $\Sigma$ and $\Sigma'$, a *lexicon* from $\Sigma$ to $\Sigma'$ is a pair $\mathscr{L} = (\sigma, \theta)$ such that

1. $\sigma$ is a type substitution that maps elements of $A$ to elements of $\mathscr{T}(A')$;
2. $\theta$ is a mapping from $C$ to $\Lambda_{\text{lin}}(\Sigma')$;
3. $\vdash_{\Sigma'} \theta(c) : \sigma(\tau(c))$ for all $c \in C$.

The mapping $\theta$ is extended to a mapping from $\Lambda_{\text{lin}}(\Sigma)$ to $\Lambda_{\text{lin}}(\Sigma')$ as follows:

$$\theta(x) = x \quad \text{for } x \in X, \qquad \theta(MN) = \theta(M)\theta(N), \qquad \theta(\lambda x.M) = \lambda x.\theta(M).$$

We write $\mathscr{L}(\alpha)$ and $\mathscr{L}(M)$ for $\sigma(\alpha)$ and $\theta(M)$, respectively. The *width* of $\mathscr{L}$ is $\max\{\, |\mathscr{L}(p)| \mid p \in A \,\}$, and the *order* of $\mathscr{L}$ is $\max\{\, \text{ord}(\mathscr{L}(p)) \mid p \in A \,\}$.

As noted by de Groote (2001), the composition of two lexicons is a lexicon: If $\mathscr{L}_1 = (\sigma_1, \theta_1)$ is a lexicon from $\Sigma_0$ to $\Sigma_1$ and $\mathscr{L}_2 = (\sigma_2, \theta_2)$ is a lexicon from $\Sigma_1$ to $\Sigma_2$, then $\mathscr{L}_2 \circ \mathscr{L}_1 = (\sigma_2 \circ \sigma_1, \theta_2 \circ \theta_1)$ is a lexicon from $\Sigma_0$ to $\Sigma_2$.

An *abstract categorial grammar* (de Groote 2001) is a quadruple $\mathscr{G} = (\Sigma, \Sigma', \mathscr{L}, s)$, where

1. $\Sigma$ is a higher-order signature called the *abstract vocabulary*;
2. $\Sigma'$ is a higher-order signature called the *object vocabulary*;
3. $\mathscr{L}$ is a lexicon from $\Sigma$ to $\Sigma'$;
4. $s$ is an element of $A$ called the *distinguished type*.

The *abstract language* of $\mathscr{G}$, denoted by $\mathcal{A}(\mathscr{G})$, and the *object language* of $\mathscr{G}$, denoted by $\mathcal{O}(\mathscr{G})$, are defined as follows:

$$\mathcal{A}(\mathscr{G}) = \{\, M \in \Lambda_{\text{lin}}(\Sigma) \mid M \text{ is in } \beta\text{-normal form and } \vdash_{\Sigma} M : s \,\}.$$
$$\mathcal{O}(\mathscr{G}) = \{\, |\mathscr{L}(M)|_{\beta} \mid M \in \mathcal{A}(\mathscr{G}) \,\}.$$

We say that an ACG *generates* its object language.

If $\Sigma'$ is a tree signature and $\mathscr{L}(s) = o$, then $\mathcal{O}(\mathscr{G})$ is a set of trees. In such a case, we call $\mathscr{G}$ a *tree-generating ACG*. If $\Sigma'$ is a string signature $\Sigma_V^{\text{string}}$ and $\mathscr{L}(s) = o \to o$, then we call $\mathscr{G}$ a *string-generating ACG*, and we say that $\mathscr{G}$ generates a string language $L \subseteq V^*$ if $\mathcal{O}(\mathscr{G}) = \{\, /w/ \mid w \in L \,\}$.

**Example 2** The following ACG $\mathscr{G} = (\Sigma, \Sigma_V^{\text{string}}, \mathscr{L}, s)$, where $V = \{a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4\}$, is constructed from the MCFG of Example 1 using de Groote and Pogodalla's (2004) procedure, and generates RESP.

$$A = \{p, q, s\}, \quad C = \{f_1, f_2, f_3, f_4, f_5\},$$
$$\tau(f_1) = p \to q \to s,\ \tau(f_2) = p,\ \tau(f_3) = q,\ \tau(f_4) = p \to p,\ \tau(f_5) = q \to q,$$
$$\mathscr{L}(p) = \mathscr{L}(q) = ((o \to o) \to (o \to o) \to o) \to o, \quad \mathscr{L}(s) = o \to o,$$
$$\mathscr{L}(f_1) = \lambda uvz.u(\lambda x_{1,1}x_{1,2}.v(\lambda x_{2,1}x_{2,2}.x_{1,1}(x_{2,1}(x_{1,2}(x_{2,2}z))))),$$
$$\mathscr{L}(f_2) = \mathscr{L}(f_3) = \lambda w.w(\lambda z.z)(\lambda z.z),$$
$$\mathscr{L}(f_4) = \lambda uw.u(\lambda x_{1,1}x_{1,2}.w(\lambda z.a_1(x_{1,1}(a_2z)))(\lambda z.a_3(x_{1,2}(a_4z)))),$$
$$\mathscr{L}(f_5) = \lambda uw.u(\lambda x_{1,1}x_{1,2}.w(\lambda z.b_1(x_{1,1}(b_2z)))(\lambda z.b_3(x_{1,2}(b_4z)))).$$

An ACG $\mathscr{G} = (\Sigma, \Sigma', \mathscr{L}, s)$ is *n-th order* if the order of $\Sigma$ does not exceed $n$. We say that $\mathscr{G}$ is *of width m* if the width of $\mathscr{L}$ does not exceed $m$. (The ACG in Example 2 is a second-order ACG of width 6.)

The notation $\mathbf{G}(n, l)$ stands for the class of $n$-th order ACGs $\mathscr{G} = (\Sigma, \Sigma', \mathscr{L}, s)$ such that the order of $\mathscr{L}$ does not exceed $l$. Kanazawa (2006a) showed that the class of string languages generated by ACGs in $\mathbf{G}(n, l)$ $(n, l \geq 2)$ is a substitution-closed full abstract family of languages, and the class of tree languages generated by ACGs in $\mathbf{G}(n, l)$ $(n \geq 2, l \geq 1)$ is closed under union, concatenation, linear non-deleting tree homomorphism, and intersection with regular tree languages. A quick examination of the proofs given in Kanazawa 2006a reveals that all constructions involved preserve the width of the lexicon as well, so the above closure properties hold of the ACGs in $\mathbf{G}(n, l)$ of width $m$.

## 1.3 Second-order ACGs for control languages

If $P$ is a linear $\lambda$-term which contains $k$ occurrences of $c$, we let $\langle P\rangle_c[y_1, \ldots, y_k]$ denote the linear $\lambda$-term that does not contain $c$ such that $\{y_1, \ldots, y_k\} \subseteq \mathrm{FV}(\langle P\rangle_c[y_1, \ldots, y_k])$ and $\langle P\rangle_c[c, \ldots, c] = P$.

The following lemma (without the condition on width) was stated by Kanazawa (2006a) without proof.

**Lemma 3** *Let $F$ be a ranked alphabet and $c$ an element of $F^{(0)}$. If $L \subseteq \mathbb{T}_F$ is generated by an ACG in $\mathbf{G}(2, l)$ of width $m$, so is $L^{*,c}$.*

*Proof.* Let $\mathscr{G} = (\Sigma, \Sigma_F^{\mathrm{tree}}, \mathscr{L}, s)$ be an ACG in $\mathbf{G}(2, l)$ of width $m$ generating $L$. We define a second-order ACG $\mathscr{G}' = (\Sigma', \Sigma_F^{\mathrm{tree}}, \mathscr{L}', s')$ generating $L^{*,c}$ as follows:

$$A' = A \cup \{s'\}, \qquad C' = C \cup \{d, e\},$$

$$\tau'(f) = \begin{cases} (s')^k \to \tau(f) & \text{if } f \in C \text{ and } c \text{ occurs } k \text{ times in } \mathscr{L}(f), \\ s \to s' & \text{if } f = d, \\ s' & \text{if } f = e, \end{cases}$$

$$\mathscr{L}'(p) = \begin{cases} \mathscr{L}(p) & \text{if } p \in A, \\ o & \text{if } p = s', \end{cases}$$

$$\mathscr{L}'(f) = \begin{cases} \lambda y_1 \ldots y_k.\langle \mathscr{L}(f)\rangle_c[y_1, \ldots, y_k] & \text{if } f \in C \text{ and } c \text{ occurs } k \text{ times in } \mathscr{L}(f), \\ \lambda x.x, & \text{if } f = d, \\ c & \text{if } f = e. \end{cases}$$

Clearly, $\mathscr{G}'$ is an ACG in $\mathbf{G}(2, l)$ of width $m$. We leave the proof of $\mathcal{O}(\mathscr{G}') = L^{*,c}$ to the reader. □

**Lemma 4** *Let $(F, h)$ be a headed ranked alphabet, and let $C \subseteq F^*$. If there is a second-order ACG of width $m$ generating $C$, then there is a second-order*

*ACG of width m generating* $\{ T \in \mathbb{T}_F \mid \text{Spines}(T) \subseteq C \}$.

*Proof.* Let $\mathscr{G}$ be a second-order ACG of width $m$ generating $C$. The idea is to first construct an ACG $\mathscr{G}'$ such that $\mathcal{O}(\mathscr{G}') = \{ \ulcorner w \urcorner \mid w \in C \}$, turn it into $\mathscr{G}''$ such that $\mathcal{O}(\mathscr{G}'') = \{ T \in \mathbb{T}_{F \cup \{c\}} \mid \text{spine}(T) \in C \text{ and } \text{iSpines}(T) \subseteq \{c\} \}$, where $c$ is a new symbol of rank 0, and then use Lemma 3 to obtain an ACG $\mathscr{G}'''$ such that $\mathcal{O}(\mathscr{G}''') = \{ T \in \mathbb{T}_F \mid \text{Spines}(T) \subseteq C \} = \mathcal{O}(\mathscr{G}'')^{*,c} \cap \mathbb{T}_F$. All constructions preserve order and width.

Let $F_0 = \{ f \in F \mid h(f) = 0 \}$, $F_1 = F - F_0$. Note that for every $T \in \mathbb{T}_F$, $\text{Spines}(T) \subseteq F_1^* F_0$. For each $f \in F_0$, let $C_f = C \cap F_1^* f$. By closure under intersection with regular sets and closure under homomorphism, we obtain from $\mathscr{G}$ an ACG generating $\{ w \mid wf \in C_f \}$. By adding to this ACG a constant $e_f$ of type $s \to s'$ whose image under the lexicon is $\lambda x.xf^{(0)}$ and changing all $g \in F_1$ to $g^{(1)}$, we obtain an ACG with distinguished type $s'$ generating the tree language $\{ \ulcorner w \urcorner \mid w \in C_f \}$. We obtain $\mathscr{G}'$ by closure under union.

To obtain $\mathscr{G}''$ from $\mathscr{G}'$, we use closure under linear non-deleting tree homomorphism. The relevant homomorphism is $\varphi$:

$$\begin{aligned} \varphi(g^{(1)}x) &= gc^{j-1}xc^{r-j}, && \text{where } r = \text{rank}(g) \text{ and } j = h(g), \\ \varphi(f^{(0)}) &= fc^r, && \text{where } r = \text{rank}(f). \end{aligned}$$

Finally, closure under Kleene star (Lemma 3) together with closure under intersection with regular sets gives $\mathscr{G}'''$. □

Let $\mathbf{A}_m$ denote the set of string languages generated by second-order ACGs of width $m$.

**Lemma 5** $\mathbf{C}_k \subseteq \mathbf{A}_{2^k}$.

*Proof.* Induction on $k$. For $k = 1$, de Groote (2001) and de Groote and Pogodalla (2004) showed that second-order ACGs of width 2 can generate all context-free languages. For the induction step, assume that $C \in \mathbf{A}_{2^k}$ and consider $L(G, C)$, where $G$ is an LDG. Since the set of complete rule trees of $G$ is a local set, by Lemma 4 and closure under intersection with regular sets, the set $\{ T \mid T \text{ is a complete rule tree of } G \text{ and } \text{Spines}(T) \subseteq C \}$ is generated by a second-order ACG of width $2^k$. Composing the lexicon of this ACG with the lexicon $\mathscr{L}_{\text{yield}_G}$ expressing $\text{yield}_G$ doubles the width ($\mathscr{L}_{\text{yield}_G}(o) = o \to o$) and gives a second-order ACG of width $2^{k+1}$ generating $L(G, C)$. □

We can show that the inclusion in Lemma 5 is proper except when $k = 1$.

**Lemma 6** $\mathbf{A}_{2^k} - \mathbf{C}_k \neq \varnothing$ *for all* $k \geq 2$.

*Proof.* For $k \geq 1$, let $\text{RESP}_k = \{ a_1^m a_2^m b_1^n b_2^n \ldots a_{2k-1}^m a_{2k}^m b_{2k-1}^n b_{2k}^n \mid m, n \geq 0 \}$. (The language RESP of Example 1 is $\text{RESP}_2$.) Palis and Shende's (1995) pumping lemma (Theorem 2) can be used to show that $\text{RESP}_{2^{k-1}} \notin \mathbf{C}_k$ for each $k \geq 2$. It is easy to define a $k$-MCFL generating $\text{RESP}_k$. While applying

de Groote and Pogodalla's (2004) procedure to a $k$-MCFG produces a second-order ACG of width $2k + 2$, this particular $k$-MCFL can be encoded in a second-order ACG of width $2k$. We omit the details for lack of space. □

## 1.4 From second-order ACGs to MCFGs

Let $V$ be an alphabet, and let $\Lambda'_{\mathrm{lin}}(\Sigma_V^{\mathrm{string}})$ be the set of linear $\lambda$-terms over $\Sigma_V^{\mathrm{string}}$ in which a symbol from $V$ always occurs in a function position. Note that $/w/ \in \Lambda'_{\mathrm{lin}}(\Sigma_V^{\mathrm{string}})$ for every $w \in V^*$. We extract from a $\lambda$-term $M \in \Lambda'_{\mathrm{lin}}(\Sigma_V^{\mathrm{string}})$ in $\beta$-normal form such that $\Gamma \vdash_{\Sigma_V^{\mathrm{string}}} M : \alpha$ a tuple $(w_1, \ldots, w_m)$ of strings over $V$ and a pure linear $\lambda$-term $P$ such that $\Gamma, z_1 : o \to o, \ldots, z_m : o \to o \vdash P : \alpha$ and $P[z_i := /w_i/]_{1 \le i \le m} \twoheadrightarrow_\beta M$. In the following definition, $\mathrm{lh}(\vec{w})$ denotes the length (i.e., number of components) of a tuple $\vec{w}$, and the symbol $\widehat{\ }$ denotes concatenation of tuples; the letters $a$ and $y$ range over symbols in $V$ and variables, respectively. We assume that for all $i \in \mathbb{N}$, $z_i \notin \mathrm{FV}(M)$.

$$\mathrm{tuple}(aM) = \begin{cases} (aw_1, w_2, \ldots, w_m) & \text{if } \mathrm{pure}(M) \text{ starts with } z_1, \\ & \quad \text{where } \mathrm{tuple}(M) = (w_1, \ldots, w_m), \\ (a)\widehat{\ }\,\mathrm{tuple}(M) & \text{otherwise,} \end{cases}$$

$$\mathrm{pure}(aM) = \begin{cases} \mathrm{pure}(M) & \text{if } \mathrm{pure}(M) \text{ starts with } z_1, \\ z_1(\mathrm{pure}(M)[z_i := z_{i+1}]_{1 \le i \le m}) & \text{otherwise, where } m = \mathrm{lh}(\mathrm{tuple}(M)), \end{cases}$$

$$\mathrm{tuple}(yM_1 \ldots M_n) = \mathrm{tuple}(M_1)\widehat{\ } \ldots \widehat{\ }\,\mathrm{tuple}(M_n),$$

$$\mathrm{pure}(yM_1 \ldots M_n) = yP_1 \ldots P_n, \quad \text{where } P_j = \mathrm{pure}(M_j)[z_i := z_{i+\sum_{k=1}^{j-1} m_k}]_{1 \le i \le m_j}, \quad m_k = \mathrm{lh}(\mathrm{tuple}(M_k)),$$

$$\mathrm{tuple}(\lambda y.M) = \mathrm{tuple}(M), \qquad \mathrm{pure}(\lambda y.M) = \lambda y.\, \mathrm{pure}(M)$$

**Lemma 7** *Let $M$ be a $\beta$-normal $\lambda$-term in $\Lambda'_{\mathrm{lin}}(\Sigma_V^{\mathrm{string}})$ such that $y_1 : \alpha_1, \ldots, y_k : \alpha_k \vdash_{\Sigma_V^{\mathrm{string}}} M : \beta$, and let $\mathrm{tuple}(M) = (w_1, \ldots, w_m)$. Then the following hold:*

(i) $y_1 : \alpha_1, \ldots, y_k : \alpha_k, z_1 : o \to o, \ldots, z_m : o \to o \vdash \mathrm{pure}(M) : \beta$.

(ii) $\mathrm{pure}(M)[z_i := /w_i/]_{1 \le i \le m} \twoheadrightarrow_\beta M$.

(iii) $m \le \dfrac{1}{2}\left(|\beta| + \displaystyle\sum_{i=1}^{k} |\alpha_i|\right)$.

*Proof.* Easy induction on $M$. For (iii), show that $2m \le |\beta| + \sum_{i=1}^{k} |\alpha_i| - 2$ if the head of $M$ is a variable. □

The rather technical lemma above may be seen as a special case of the main result concerning interpolation of Kanazawa (2006b).

Let $\mathscr{G} = (\Sigma, \Sigma_V^{\mathrm{string}}, \mathscr{L}, s)$ be a second-order string-generating ACG of width $m$. We assume that for every $c \in A$, $\mathscr{L}(c)$ belongs to $\Lambda'_{\mathrm{lin}}(\Sigma_V^{\mathrm{string}})$. For $\alpha \in \mathscr{T}(\{o\})$, let $\mathbf{P}_\alpha$ be the set of pure linear $\lambda$-terms $M$ in $\beta$-normal form such

that $z_1 : o \to o, \ldots, z_r : o \to o \vdash M : \alpha$ for some $r \leq \frac{1}{2}|\alpha|$. Note that $\mathbf{P}_\alpha$ is finite. Define an MCFG $\mathrm{mcfg}(\mathscr{G}) = (N, V, P, S)$ as follows:

- $N = \{S\} \cup \{\, (p, M) \mid p \in A, M \in \mathbf{P}_{\mathscr{L}(p)} \,\}$, where $\mathrm{rank}((p, M)) = |\mathrm{FV}(M)|$.
- For each $c \in C$ with $\tau(c) = p_1 \to \cdots \to p_n \to p$ and $M_i \in \mathbf{P}_{\mathscr{L}(p_i)}$ with $|\mathrm{FV}(M_i)| = r_i$ $(1 \leq i \leq n)$, $P$ contains the rule

  $$(p, \mathrm{pure}(M))(\mathrm{tuple}(M)) :- (p_1, M_1)(x_{1,1}, \ldots, x_{1,r_1}), \ldots, (p_n, M_n)(x_{n,1}, \ldots, x_{n,r_n})$$

  where $M = |\mathscr{L}(c)(M_1[z_j := /x_{1,j}/]_{1 \leq j \leq r_1}) \ldots (M_n[z_j := /x_{n,j}/]_{1 \leq j \leq r_n})|_\beta$ and $\mathrm{tuple}(M)$, $\mathrm{pure}(M)$ are defined with respect to the alphabet $V \cup \{\, x_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq r_i \,\}$.
- In addition, $P$ contains the following rules:

  $$S(\epsilon) :- (s, \lambda z.z). \qquad S(x_{1,1}) :- (s, \lambda z.z_1 z)(x_{1,1}).$$

**Lemma 8** *If $\mathscr{G}$ is a second-order ACG of width $m$ generating $L \subseteq V^*$, then* $\mathrm{mcfg}(\mathscr{G})$ *is an $\lfloor \frac{1}{2}m \rfloor$-MCFG and $L(\mathrm{mcfg}(\mathscr{G})) = L$. Hence $\mathbf{A}_m \subseteq \lfloor \frac{1}{2}m \rfloor$-MCFL.*

**Theorem 9** $\mathbf{C}_k \subseteq 2^{k-1}$-MCFL. *The inclusion is proper for $k \geq 2$.*

*Proof.* From Lemmas 5, 6, and 8. □

## 1.5 Conclusion

We have proved $\mathbf{C}_k \subsetneq 2^{k-1}$-MCFL for each $k \geq 2$ by showing the following two inclusions:

$$\mathbf{C}_k \subsetneq \mathbf{A}_{2^k} \subseteq 2^{k-1}\text{-MCFL}.$$

We do not know whether the second inclusion is proper. Note that by de Groote and Pogodalla's (2004) result, we know $2^{k-1}$-MCFL $\subseteq \mathbf{A}_{2^k+2}$. Another outstanding open question is whether the inclusion of $\bigcup_{k \geq 1} \mathbf{C}_k$ in MCFL is proper.

## References

Comon, Hubert, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. 2002. *Tree Automata Techniques and Applications*. Available online at `http://www.grappa.univ-lille3.fr/tata/`. Latest version dated September 6, 2005.

de Groote, Philippe. 2001. Towards abstract categorial grammars. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 252–259.

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.

Gécseg, Ferenc and Magnus Steinby. 1997. Tree languages. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages, Vol. 3: Beyond Words*, pages 1–68. Berlin: Springer.

Groenink, Annius V. 1997. *Surface without Structure*. Ph.D. thesis, University of Utrecht.

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

Kanazawa, Makoto. 2006a. Abstract families of abstract categorial languages. *Electronic Notes in Theoretical Computer Science* 165:65–80.

Kanazawa, Makoto. 2006b. Computing interpolants in implicational logics. *Annals of Pure and Applied Logic* 142(1–3):125–201.

Kasami, Tadao, Hiroyuki Seki, and Mamoru Fujii. 1987. Generalized context-free grammars, multiple context-free grammars and head grammars. Tech. rep., Osaka University.

Palis, Michael A. and Sunil M. Shende. 1992. Upper bounds on recognition of a hierarchy of non-context-free languages. *Theoretical Computer Science* 98(2):289–319.

Palis, M. A. and S. M. Shende. 1995. Pumping lemmas for the control language hierarchy. *Mathematical Systems Theory* 28(3):199–213.

Radzinski, Daniel. 1991. Chinese number-names, tree adjoining languages, and mild context-sensitivity. *Computational Linguistics* 17(3):277–299.

Salvati, Sylvain. 2007. Encoding second order string ACG with deterministic tree walking transducers. In S. Wintner, ed., *Proceedings of FG 2006: The 11th conference on Formal Grammar*, FG Online Proceedings, pages 143–156. CSLI Publications.

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

Sørensen, Morten Heine and Paweł Urzyczyn. 2006. *Lectures on the Curry-Howard Isomorphism*. Amsterdam: Elsevier.

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

Vijay-Shanker, K., David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Weir, David J. 1992a. A geometric hierarchy beyond context-free languages. *Theoretical Computer Science* 104(2):235–261.

Weir, David J. 1992b. Linear context-free rewriting systems and deterministic tree-walking transducers. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 136–143.