Abstract Families of Abstract Categorial Languages

Makoto Kanazawa [1]

National Institute of Informatics
Tokyo, Japan

Abstract

We show that the class of string languages generated by abstract categorial grammars is a substitution-closed full AFL. The result also holds of each class $\mathbf{G}(m, n)$ in de Groote's hierarchy. We also show that the class of string languages generated by lexicalized ACGs is a substitution-closed AFL, and that most of the results about string languages carry over to tree languages.

Key words: Abstract Categorial Grammar, Abstract Family of Languages, Tree Languages.

1 Introduction

The *abstract categorial grammar* (ACG, [8]) elegantly generalizes and unifies diverse types of grammar formalisms that have been proposed for the description of natural language, including both string grammars and tree grammars. ACGs are formalized in terms of linear lambda calculus, and they generate languages of linear lambda terms, of which usual string languages and tree languages are two special cases. Although some important results have been obtained about the expressive power of ACGs ([9,11,19,16]), little is known about the entire class of (string/tree) languages generated by ACGs; in particular, no example of an r.e. language has been found which lies outside of this class.

The main result of this paper is that the string languages generated by ACGs form a substitution-closed *full AFL* (*abstract family of languages*) in the sense of Ginsburg and Greibach [6]. This result also holds of each class $\mathbf{G}(m, n)$ in de Groote's hierarchy. The string languages of *lexicalized* ACGs, which cannot contain the empty string, can be shown to satisfy somewhat

[1] I am grateful to Philippe de Groote for suggesting a generalization of Lemma 4.1 which led to the present formulation of Lemma 3.3.

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

weaker conditions, forming a substitution-closed *AFL*. We also show how most of these results can be generalized from string languages to tree languages.

The results in this paper are interesting for at least three reasons. First, in the absence of an automaton model for ACGs, they are by no means obvious. In fact, the proof of closure under intersection with regular sets is an interesting application of the Curry-style type assignment system $\lambda{\rightarrow}$. Second, these results will hopefully be useful for gaining insights into the open questions about the generative capacity and complexity of ACGs. For instance, in the event that an analogue of the Pumping Lemma becomes available for ACGs, closure under intersection with regular sets will surely be helpful in showing many languages to fall outside of the class of ACG languages. Third, since the closure properties of full AFLs are consequences of the existence of an appropriate kind of nondeterministic acceptor, the fact that the string languages of ACGs form a full AFL suggests the possibility of a natural automaton model corresponding to ACGs.

2 Preliminaries

This section presents some concepts and results which will be necessary in the sequel. The reader may consult [6,13,8] for more details.

2.1 Abstract Families of Languages

We take for granted standard language-theoretic notions like *concatenation*, the *Kleene star* ($*$) and *Kleene plus* ($+$) operations, *homomorphism*, etc. A *family of languages* is a class of languages (not necessarily over the same alphabet) which contains at least one non-empty language. A homomorphism h from V_1^* to V_2^*, where V_1 and V_2 are finite alphabets, is *ϵ-free* if $h(w) = \epsilon$ implies $w = \epsilon$. A family $\mathscr{F}$ of languages is *closed under inverse homomorphism* if whenever $L \subseteq V_1^*$ is a member of $\mathscr{F}$ and h is a homomorphism from V_2^* to V_1^*, $h^{-1}(L) = \{\, w \in V_2^* \mid h(w) \in L \,\}$ is in $\mathscr{F}$.

A mapping f from an alphabet V_1 to $\mathscr{P}(V_2^*)$, for some alphabet V_2, is called a *substitution*. The mapping f is extended to strings and to languages as follows:

$$f(\epsilon) = \{\epsilon\}, \qquad f(wa) = f(w)f(a),$$
$$f(L) = \bigcup_{w \in L} f(w).$$

A family $\mathscr{F}$ of languages is *closed under substitution* if, for every $L \in \mathscr{F}$ such that $L \subseteq V_1^*$ and for every substitution f from V_1 to $\mathscr{P}(V_2^*)$ such that $f(a) \in \mathscr{F}$ for all $a \in V_1$, we have $f(L) \in \mathscr{F}$.

Let $\mathscr{F}$ be a family of languages. We say that $\mathscr{F}$ is an *AFL* (*abstract family of languages*) if $\mathscr{F}$ is closed under union, concatenation, Kleene plus, ϵ-free homomorphism, inverse homomorphism, and intersection with regular

sets. We say that $\mathscr{F}$ is a *full AFL* if $\mathscr{F}$ is closed under union, concatenation, Kleene star, homomorphism, inverse homomorphism, and intersection with regular sets.

Some well-known examples of full AFLs are the regular sets, the context-free languages, the r.e. sets, the indexed languages ([1]), the linear indexed languages ([4]), the multiple context-free languages and the parallel multiple context-free languages ([17]). Some examples of AFLs that are not full AFLs are the context-sensitive languages, the recursive sets, the ϵ-free context-free languages, and the class NP. An example of a family of languages that is not an AFL is the class of PTIME languages (assuming P $\neq$ NP).

Many types of grammars that are known to generate full AFLs have a corresponding type of nondeterministic acceptor. In such cases, closure under the regular operations becomes easy to prove. Proof of the other closure properties is also not hard, given the following fact ([6]):

Fact. *A family of languages is closed under homomorphism, inverse homomorphism, and intersection with regular sets if and only if it is closed under finite transduction.*

2.2 Type Assignment System $\lambda{\rightarrow}_{\Sigma}$

Given a finite set A of *atomic types*, the set $\mathscr{T}(A)$ of *types* built upon A is the smallest superset of A satisfying the following condition:

$$\alpha, \beta \in \mathscr{T}(A) \quad \text{implies} \quad (\alpha \to \beta) \in \mathscr{T}(A).$$

We omit the outermost parentheses when we write types. The connective $\to$ is assumed to be right-associative, so we write $\alpha_1 \to \alpha_2 \to \alpha_3$ instead of $\alpha_1 \to (\alpha_2 \to \alpha_3)$. We abbreviate $\underbrace{\alpha \to \cdots \to \alpha}_{k \text{ times}} \to \beta$ as $\alpha^k \to \beta$. We define $\mathrm{arity}(\alpha) = k$ if $\alpha = \alpha_1 \to \cdots \to \alpha_k \to p$ for some $p \in A$.

The *order* of a type α, denoted by $\mathrm{ord}(\alpha)$, is defined as follows:

$$\begin{aligned} \mathrm{ord}(p) &= 1 \quad \text{if } p \text{ is atomic,} \\ \mathrm{ord}(\alpha \to \beta) &= \max(\mathrm{ord}(\alpha) + 1, \mathrm{ord}(\beta)). \end{aligned}$$

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

(i) If $M, N \in \Lambda(\Sigma)$, then $(MN) \in \Lambda(\Sigma)$;

(ii) If $M \in \Lambda(\Sigma)$ and $x \in X$, then $(\lambda x.M) \in \Lambda(\Sigma)$.

We omit the outermost parentheses when we write λ-terms. We

write MNP for $(MN)P$, $\lambda x.MN$ for $\lambda x.(MN)$, and $\lambda x_1 \dots x_n.M$ for $\lambda x_1.(\lambda x_2.\dots(\lambda x_n.M)\dots)$. The set $\mathrm{FV}(M)$ of *free variables* of M is understood in the usual way. We write $M[x_1,\dots,x_n]$ to indicate that $\{x_1,\dots,x_n\} \subseteq \mathrm{FV}(M[x_1,\dots,x_n])$. A λ-term M is *closed* if $\mathrm{FV}(M) = \varnothing$; it is a *combinator* if it moreover contains no constants. We take for granted the notions of *substitution* (of a λ-term for a free variable in a λ-term), *β-redex*, *β-reduction*, *β-normal form*, etc. We write $\twoheadrightarrow_\beta$ for β-reduction, and $=_\beta$ for β-equality. The β-normal form of M is denoted by $|M|_\beta$.

Let $M \twoheadrightarrow_\beta M'$ in one step by contraction of a β-redex $(\lambda x.N)P$. This β-reduction step is said to be *non-erasing* if $x \in \mathrm{FV}(N)$. It is said to be *non-duplicating* if x occurs free in N at most once. The β-reduction from M to M' is non-erasing (non-duplicating) if it consists entirely of non-erasing (non-duplicating) β-reduction steps.

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

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

$$\frac{\Gamma \vdash_\Sigma M : \beta}{\Gamma - \{x:\alpha\} \vdash_\Sigma \lambda x.M : \alpha \to \beta} \qquad\qquad \frac{\Gamma \vdash_\Sigma M : \alpha \to \beta \quad \Delta \vdash_\Sigma N : \alpha}{\Gamma \cup \Delta \vdash_\Sigma MN : \beta}$$

We write $\Gamma \vdash M : \alpha$ when $\Gamma \vdash_\Sigma M : \alpha$ for some $\Sigma = \langle A, \varnothing, \varnothing\rangle$. As usual, we have $\Gamma \vdash_\Sigma M : \alpha$ if and only if there is a *$\lambda{\to_\Sigma}$-deduction* of this judgment. M is *typable* if $\Gamma \vdash_\Sigma M : \alpha$ for some Γ, α.

A λ-term M is *linear* if the following conditions both hold:

(i) for any subterm $\lambda x.N$ of M, $x \in \mathrm{FV}(M)$;

(ii) for any subterm NP of M, $\mathrm{FV}(N) \cap \mathrm{FV}(P) = \varnothing$.

M is a *λI-term* if it satisfies the first condition.

It is known that a linear λ-term which contains no constants is always typable ([12]). The set of linear λ-terms over Σ is denoted $\Lambda_{\mathrm{lin}}(\Sigma)$.

Below we list some important facts about $\lambda{\to_\Sigma}$ which we will make use of in this paper (see [13]).

Subject Reduction Theorem. *If $\Gamma \vdash_\Sigma M : \alpha$ and $M \twoheadrightarrow_\beta M'$, then $\Gamma' \vdash_\Sigma M' : \alpha$, where Γ' is the restriction of Γ to $\mathrm{FV}(M')$.*

Subject Expansion Theorem. *If $\Gamma \vdash_\Sigma M' : \alpha$ and $M \twoheadrightarrow_\beta M'$ by non-erasing non-duplicating β-reduction, then $\Gamma \vdash_\Sigma M : \alpha$.*

As a special case, if M is linear and $M \twoheadrightarrow_\beta M'$, then $\Gamma \vdash_\Sigma M' : \alpha$ implies $\Gamma \vdash_\Sigma M : \alpha$.

Uniqueness Theorem. *If M is a λI-term and $\Gamma \vdash_\Sigma M : \alpha$, then there is a unique $\lambda{\to_\Sigma}$-deduction of this judgment.*

A pair $\langle \Gamma, \alpha \rangle$ is a *principal pair* for M if $\Gamma \vdash M : \alpha$ and for every Γ', M' such that $\Gamma' \vdash M : \alpha'$, there is a type substitution σ such that $\Gamma' = \sigma(\Gamma)$ and $\alpha' = \sigma(\alpha)$.

Principal Pair Theorem. *If M is typable, then there is a principal pair for M.*

2.3 Trees and Strings as Linear λ-terms

A *ranked alphabet* is a pair $\langle F, \rho \rangle$, where F is an alphabet and ρ is a mapping $\rho\colon F \to \mathbb{N}$. If $\langle F, \rho \rangle$ is a ranked alphabet, we write F_k for $\{\, f \in F \mid \rho(f) = k \,\}$. We often write F for the ranked alphabet $\langle F, \rho \rangle$, suppressing reference to ρ. The set of trees over a ranked alphabet F, denoted by $\mathbb{T}_F$, is the smallest superset of F_0 that satisfies the following condition:

$$f \in F_k \text{ and } T_1, \ldots, T_k \in \mathbb{T}_F \quad \text{implies} \quad (fT_1 \ldots T_k) \in \mathbb{T}_F.$$

A set of trees is called a *tree language*. We refer the reader to [3] or [5] for basic concepts about tree languages.

A ranked alphabet F can be represented by a second-order signature $\Sigma_F = \langle \{o\}, F, \tau_F \rangle$, where for each $f \in F_k$, $\tau_F(f) = o^k \to o$. Σ_F is called a *tree signature*. We identify a tree in $\mathbb{T}_F$ with a closed λ-term in $\Lambda_{\text{lin}}(\Sigma_F)$ of type o in the obvious way.

A string $a_1 \ldots a_n$ over an (unranked) alphabet V can be represented by a closed λ-term

$$/a_1 \ldots a_n/ = \lambda z.a_1(\ldots(a_n z)\ldots)$$

in $\Lambda_{\text{lin}}(\Sigma_V)$, where $\Sigma_V = \langle \{o\}, V, \tau \rangle$ and $\tau(a) = o \to o$ for all $a \in V$. We call Σ_V a *string signature*. Note that $\vdash_{\Sigma_V} /w/ : o \to o$ for all strings $w \in V^*$.

2.4 Abstract Categorial Grammars

When we write $\Sigma, \Sigma', \Sigma_1$, etc., to refer to higher-order signatures, we assume $\Sigma = \langle A, C, \tau \rangle$, $\Sigma' = \langle A', C', \tau' \rangle$, $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$, etc., unless otherwise noted. Given higher-order signatures Σ and Σ', a *lexicon* from Σ to Σ' is a pair $\mathscr{L} = \langle \sigma, \theta \rangle$ such that

(i) σ is a type substitution that maps elements of A to elements of $\mathscr{T}(A')$;

(ii) θ is a mapping from C to $\Lambda_{\text{lin}}(\Sigma')$;

(iii) $\vdash_{\Sigma'} \theta(c) : \sigma(\tau(c))$ for all $c \in C$.

θ 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 *order* of $\mathscr{L}$ is $\max\{\, \text{ord}(\mathscr{L}(p)) \mid p \in A \,\}$.

We list three easy properties of lexicons. First, typing judgments are preserved under lexicons: If $\mathscr{L}$ is a lexicon from Σ to Σ', then

$$\Gamma \vdash_\Sigma M : \alpha \quad \text{implies} \quad \mathscr{L}(\Gamma) \vdash_{\Sigma'} \mathscr{L}(M) : \mathscr{L}(\alpha).$$

Second, β-reduction commutes with lexicons:

$$M \twoheadrightarrow_\beta M' \quad \text{implies} \quad \mathscr{L}(M) \twoheadrightarrow_\beta \mathscr{L}(M').$$

Third, the composition of two lexicons is a lexicon: If $\mathscr{L}_1 = \langle \sigma_1, \theta_1 \rangle$ is a lexicon from Σ_0 to Σ_1 and $\mathscr{L}_2 = \langle \sigma_2, \theta_2 \rangle$ is a lexicon from Σ_1 to Σ_2, then

$$\mathscr{L}_2 \circ \mathscr{L}_1 = \langle \sigma_2 \circ \sigma_1, \theta_2 \circ \theta_1 \rangle$$

is a lexicon from Σ_0 to Σ_2.

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

(i) Σ is a higher-order signature called the *abstract vocabulary*;

(ii) Σ' is a higher-order signature called the *object vocabulary*;

(iii) $\mathscr{L}$ is a lexicon from Σ to Σ';

(iv) s is an atomic type of the abstract vocabulary ($s \in A$).

The *abstract language* of $\mathscr{G}$, denoted by $\mathcal{A}(\mathscr{G})$, is defined as follows:

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

The *object language* of $\mathscr{G}$, denoted by $\mathcal{O}(\mathscr{G})$, is defined as follows:

$$\mathcal{O}(\mathscr{G}) = \{\, |\mathscr{L}(M)|_\beta \mid M \in \mathcal{A}(\mathscr{G}) \,\}.$$

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

If Σ' 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 ACG*. If Σ' is a string signature Σ_V and $\mathscr{L}(s) = o \to o$, then we call $\mathscr{G}$ a *string ACG*, and we say that $\mathscr{G}$ generates a string language $L \subseteq V^*$ if $\mathcal{O}(\mathscr{G}) = \{\, /w/ \mid w \in L \,\}$.

A constant c of the abstract vocabulary of an ACG $\mathscr{G}$ is *lexical* if $\mathscr{L}(c)$ is not a combinator, i.e., if it contains at least one constant. We say that $\mathscr{G}$ is *lexicalized* (in analogy with *lexicalized tree-adjoining grammars* ([14])) if all its abstract constants are lexical. We denote the class of lexicalized ACGs by **Lex**.

For $m \geq 2, n \geq 1$, $\mathbf{G}(m, n)$ denotes the class of ACGs $\mathscr{G} = \langle \Sigma, \Sigma', \mathscr{L}, s \rangle$ such that the order of Σ is $\leq m$ and the order of $\mathscr{L}$ is $\leq n$. An ACG $\mathscr{G}$ is *m-th order* if $\mathscr{G} \in \mathbf{G}(m, n)$ for some n. Note that if $\mathscr{G} \in \mathbf{G}(m, n)$ is a string ACG, $n \geq 2$.

Example 2.1 Let $\mathscr{G} = \langle \Sigma, \Sigma', \mathscr{L}, s \rangle$, where

$$
\begin{aligned}
A &= \{p_1, p_2, p_3, q, s\}, & A' &= \{o\},\\
C &= \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}\}, & C' &= \{\mathtt{a}, \mathtt{b}, \mathtt{c}\},\\
\tau(\mathrm{A}) &= (p_1 \to s) \to s, & \tau'(\mathtt{a}) &= o \to o,\\
\tau(\mathrm{B}) &= (p_2 \to s) \to s, & \tau'(\mathtt{b}) &= o \to o,\\
\tau(\mathrm{C}) &= (p_3 \to s) \to s, & \tau'(\mathtt{c}) &= o \to o,\\
\tau(\mathrm{D}) &= q \to s,\\
\tau(\mathrm{E}) &= p_1 \to p_2 \to p_3 \to q \to q,\\
\tau(\mathrm{F}) &= q,
\end{aligned}
$$

$$
\begin{aligned}
\mathscr{L}(p_1) &= o \to o, & \mathscr{L}(\mathrm{A}) &= \lambda u.\mathtt{a}(u(\lambda z.z)),\\
\mathscr{L}(p_2) &= o \to o, & \mathscr{L}(\mathrm{B}) &= \lambda u.\mathtt{b}(u(\lambda z.z)),\\
\mathscr{L}(p_3) &= o \to o, & \mathscr{L}(\mathrm{C}) &= \lambda u.\mathtt{c}(u(\lambda z.z)),\\
\mathscr{L}(q) &= o \to o, & \mathscr{L}(\mathrm{D}) &= \lambda v.v,\\
\mathscr{L}(s) &= o \to o, & \mathscr{L}(\mathrm{E}) &= \lambda x_1 x_2 x_3 v z.x_1(x_2(x_3(vz))),\\
& & \mathscr{L}(\mathrm{F}) &= \lambda z.z.
\end{aligned}
$$

Then $\mathscr{G} \in \mathbf{G}(3,2)$. Let

$$P = \mathrm{A}(\lambda x_1.\mathrm{B}(\lambda y_1.\mathrm{B}(\lambda y_2.\mathrm{A}(\lambda x_2.\mathrm{C}(\lambda z_1.\mathrm{C}(\lambda z_2.\mathrm{D}(\mathrm{E}x_1y_1z_1(\mathrm{E}x_2y_2z_2\mathrm{F}))))))))).$$

We have $P \in \mathcal{A}(\mathscr{G})$ and $|\mathscr{L}(P)|_\beta = /\mathtt{abbacc}/$, so $/\mathtt{abbacc}/ \in \mathcal{O}(\mathscr{G})$. It is not hard to see $\mathcal{O}(\mathscr{G}) = \{\, /w/ \mid w \in \textsc{mix} \,\}$, where

$$\textsc{mix} = \{\, w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid \mathtt{a},\ \mathtt{b},\ \mathtt{c} \text{ occur in } w \text{ the same number of times} \,\}.$$

It is not yet known whether the universal membership problem "$M \in \mathcal{O}(\mathscr{G})$?" for ACGs is decidable. It is known ([10,19]) that this problem is at least EXPSPACE-hard. Restricted to the class of lexicalized ACGs, the universal membership problem becomes NP-complete ([16,19]).

As for the generative capacity of ACGs, no r.e. set has been found that cannot be generated by any ACGs. The languages generated by lexicalized ACGs form a subset of NP that contains some NP-complete languages ([16,19]).

It has been shown that many well-known grammar formalisms can be encoded by second-order ACGs in a direct way ([8,9,11]). The tree languages generated by ACGs in $\mathbf{G}(2,1)$ are exactly the regular tree languages, and the string languages generated by ACGs in $\mathbf{G}(2,2)$ are exactly the context-free languages. De Groote and Pogodalla [11] prove that the string languages generated by ACGs in $\mathbf{G}(2,3)$ include those generated by *linear non-deleting context-free tree grammars*, and the method employed there also shows that the tree languages generated by ACGs in $\mathbf{G}(2,2)$ include those generated by linear non-deleting context-free tree grammars (see [15] for a complete proof). De Groote and Pogodalla [11] also show that ACGs in $\mathbf{G}(2,4)$ can encode *linear context-free rewriting systems* (LCFRSs, [18]), so that the string languages

generated by the latter are included in those generated by the former.[2] It is easy to see that the string languages generated by second-order ACGs are *semilinear*. Salvati [16] shows that the object language of any second-order ACG is in PTIME.[3]

3 General closure properties

In this section, we prove some properties of the languages generated by arbitrary ACGs, not just string or tree ACGs.

Lemma 3.1 (Composition with a lexicon) *Let* $\mathscr{G}_1 = \langle \Sigma_0, \Sigma_1, \mathscr{L}_1, s \rangle \in \mathbf{G}(m,n)$*, and let* $\mathscr{L}_2$ *be a* k*-th order lexicon from* Σ_1 *to* Σ_2*. Then* $\mathscr{G}_2 = \langle \Sigma_0, \Sigma_2, \mathscr{L}_2 \circ \mathscr{L}_1, s \rangle \in \mathbf{G}(m, n+k-1)$ *and* $\mathcal{O}(\mathscr{G}_2) = \{\, |\mathscr{L}_2(N)|_\beta \mid N \in \mathcal{O}(\mathscr{G}_1) \,\}$.

Let $\Sigma = \langle A, C, \tau \rangle$ be a higher-order signature. Let $c \in C$ and let $L \subseteq \{\, N \in \Lambda_{\mathrm{lin}}(\Sigma) \mid \vdash_\Sigma N : \tau(c) \,\}$. For $M \in \Lambda_{\mathrm{lin}}(\Sigma)$, we define $M[c \leftarrow L]$ inductively as follows:

$$\begin{aligned}
x[c \leftarrow L] &= \{x\} \quad \text{for } x \in X, \\
d[c \leftarrow L] &= \{d\} \quad \text{for } d \in C \text{ with } d \neq c, \\
c[c \leftarrow L] &= L, \\
(MN)[c \leftarrow L] &= \{\, PQ \mid P \in M[c \leftarrow L] \text{ and } Q \in N[c \leftarrow L] \,\}, \\
(\lambda x.M)[c \leftarrow L] &= \{\, \lambda x.P \mid P \in M[c \leftarrow L] \,\}.
\end{aligned}$$

If $L' \subseteq \Lambda_{\mathrm{lin}}(\Sigma)$, we define $L'[c \leftarrow L]$ by

$$L'[c \leftarrow L] = \bigcup_{M \in L'} M[c \leftarrow L].$$

Lemma 3.2 (Closure under substitution) *Let* $\mathscr{G}_1 = \langle \Sigma_1, \Sigma_{\mathrm{obj}}, \mathscr{L}_1, s_1 \rangle$ *and* $\mathscr{G}_2 = \langle \Sigma_2, \Sigma_{\mathrm{obj}}, \mathscr{L}_2, s_2 \rangle$ *be in* $\mathbf{G}(m,n)$*. Suppose that* $c \in C_{\mathrm{obj}}$ *and* $\tau_{\mathrm{obj}}(c) = \mathscr{L}_2(s_2)$*. Then there is a* $\mathscr{G} \in \mathbf{G}(m,n)$ *such that* $\mathcal{O}(\mathscr{G}) = \{\, |P|_\beta \mid P \in \mathcal{O}(\mathscr{G}_1)[c \leftarrow \mathcal{O}(\mathscr{G}_2)] \,\}$*. Moreover, if* $\mathscr{G}_1, \mathscr{G}_2 \in \mathbf{Lex}$*, then* $\mathscr{G} \in \mathbf{Lex}$.

Proof. Without loss of generality, we assume that $A_1 \cap A_2 = \varnothing$ and $C_1 \cap C_2 = \varnothing$. Let

$$A = A_1 \cup A_2, \quad C = C_1 \cup C_2.$$

Define $\tau \colon C \to \mathscr{T}(A)$ and a lexicon $\mathscr{L}$ from $\Sigma = \langle A, C, \tau \rangle$ to Σ_{obj} as follows.

[2] Seki et al. [17] show that the family of multiple context-free languages coincides with the family of languages generated by LCFRSs.

[3] As for third-order ACGs, Yoshinaka and Kanazawa [19] show that ACGs in $\mathbf{G}(3,2)$ can generate string languages that are not semilinear, and Salvati [16] shows that ACGs in $\mathbf{G}(3,1)$ can generate NP-complete tree languages.

Let

$$\mathscr{L}(p) = \begin{cases} \mathscr{L}_1(p) & \text{if } p \in A_1, \\ \mathscr{L}_2(p) & \text{if } p \in A_2. \end{cases}$$

For $d \in C_1$, if $\mathscr{L}_1(d)$ has k occurrences of c, let

$$\tau(d) = s_2^k \to \tau_1(d), \quad \mathscr{L}(d) = \lambda y_1 \dots y_k . P_d[y_1, \dots, y_k],$$

where $P_d[y_1, \dots, y_k] \in \Lambda_{\mathrm{lin}}(\Sigma_{\mathrm{obj}})$ and $P_d[c, \dots, c] = \mathscr{L}_1(d)$. For $d \in C_2$, let

$$\tau(d) = \tau_2(d), \quad \mathscr{L}(d) = \mathscr{L}_2(d).$$

Then $\mathscr{G} = \langle \Sigma, \Sigma_{\mathrm{obj}}, \mathscr{L}, s_1 \rangle \in \mathbf{G}(m, n)$ and it is not difficult to see $\mathcal{O}(\mathscr{G}) = \{\, |P|_\beta \mid P \in \mathcal{O}(\mathscr{G}_1)[c \leftarrow \mathcal{O}(\mathscr{G}_2)] \,\}$.

To make $\mathscr{G}$ lexicalized when $\mathscr{G}_1$ and $\mathscr{G}_2$ are, modify the construction as follows. If $d \in C_1$ is non-lexical in $\mathscr{G}$, replace it with the following set of constants

$$\{\, d_e \mid e \in C_2 \text{ and } \tau_2(e) \text{ ends in } s_2 \,\},$$

and let

$$\begin{aligned} \tau(d_e) &= s_2^{k-1} \to \alpha_1 \to \dots \to \alpha_l \to \tau_1(d), \\ \mathscr{L}(d_e) &= \lambda y_2 \dots y_k z_1 \dots z_l . P_d[\mathscr{L}_2(e) z_1 \dots z_l, y_2, \dots, y_k], \end{aligned}$$

where $\tau_2(e) = \alpha_1 \to \dots \to \alpha_l \to s_2$, and k and $P_d[y_1, \dots, y_k]$ are as above. □

A lexicon $\mathscr{L}$ from Σ to Σ' is called a *relabeling* if $\mathscr{L}(c) \in C'$ for all $c \in C$ and $\mathscr{L}(p) \in A'$ for all $p \in A$. If $L \subseteq \Lambda_{\mathrm{lin}}(\Sigma')$, we let $\mathscr{L}^{-1}(L)$ denote $\{\, M \in \Lambda_{\mathrm{lin}}(\Sigma') \mid \mathscr{L}(M) \in L \,\}$.

Lemma 3.3 *Let $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$ be an ACG in $\mathbf{G}(m, n)$ and $\mathscr{L}_1$ be a relabeling from Σ_1' to Σ_1. Let γ be a type in $\mathscr{T}(A_1')$ such that $\mathscr{L}_1(\gamma) = \mathscr{L}(s)$. Then one can find an ACG $\mathscr{G}'$ in $\mathbf{G}(m, n)$ such that*

$$\mathcal{O}(\mathscr{G}') = \{\, M \in \Lambda_{\mathrm{lin}}(\Sigma_1') \mid M \textit{ is } \beta\textit{-normal and } \vdash_{\Sigma_1'} M : \gamma \,\} \cap \mathscr{L}_1^{-1}(\mathcal{O}(\mathscr{G})).$$

Moreover, if $\mathscr{G} \in \mathbf{Lex}$, then $\mathscr{G}' \in \mathbf{Lex}$.

Proof. Define a signature Σ_0' by

$$\begin{aligned} A_0' &= \{\, p^\beta \mid p \in A_0, \beta \in \mathscr{T}(A_1'), \mathscr{L}_1(\beta) = \mathscr{L}(p) \,\}, \\ C_0' &= \{\, d_{\langle c, N, \beta \rangle} \mid c \in C_0, N \in \Lambda_{\mathrm{lin}}(\Sigma_1'), \beta \in \mathscr{T}(A_1'), \\ &\qquad \mathscr{L}_1(N) = \mathscr{L}(c), \mathscr{L}_1(\beta) = \mathscr{L}(\tau(c)), \text{ and } \vdash_{\Sigma_1'} N : \beta \,\}, \\ \tau_0'(d_{\langle c, N, \beta \rangle}) &= \mathrm{anti}(\tau(c), \beta), \end{aligned}$$

where

$$\mathrm{anti}(p,\beta) = p^{\beta}, \quad \mathrm{anti}(\alpha_1 \to \alpha_2, \beta_1 \to \beta_2) = \mathrm{anti}(\alpha_1,\beta_1) \to \mathrm{anti}(\alpha_2,\beta_2).$$

Note that $\tau_0'(d_{\langle c,N,\beta\rangle})$ is always defined and is a *most specific common anti-instance* of $\tau(c)$ and β.

Define a lexicon $\mathscr{L}_0 = \langle \sigma_0, \theta_0 \rangle$ from Σ_0' to Σ_0 and a lexicon $\mathscr{L}' = \langle \sigma', \theta' \rangle$ from Σ_0' to Σ_1' as follows:

$$\begin{aligned} \sigma_0(p^{\beta}) &= p, & \sigma'(p^{\beta}) &= \beta, \\ \theta_0(d_{\langle c,N,\beta\rangle}) &= c, & \theta'(d_{\langle c,N,\beta\rangle}) &= N. \end{aligned}$$

We have

$$\mathscr{L} \circ \mathscr{L}_0 = \mathscr{L}_1 \circ \mathscr{L}',$$

as is depicted in the following diagram:

$$\begin{array}{ccc} \vdash_{\Sigma_0} c : \tau(c) & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \mathscr{L}(c) : \mathscr{L}(\tau(c)) \\ \uparrow \mathscr{L}_0 & & \uparrow \mathscr{L}_1 \\ \vdash_{\Sigma_0'} d_{\langle c,N,\beta\rangle} : \mathrm{anti}(\tau(c),\beta) & \xrightarrow[\mathscr{L}']{} & \vdash_{\Sigma_1'} N : \beta \end{array}$$

Let $\mathscr{G}' = \langle \Sigma_0', \Sigma_1', \mathscr{L}', s^{\gamma} \rangle$. It is easy to see $\mathscr{G}' \in \mathbf{G}(m,n)$, and $\mathscr{G}' \in \mathbf{Lex}$ if $\mathscr{G} \in \mathbf{Lex}$.

Claim. $\mathcal{O}(\mathscr{G}') \subseteq \{ M \in \Lambda_{\mathrm{lin}}(\Sigma_1') \mid M \textit{ is } \beta\textit{-normal and } \vdash_{\Sigma_1'} M : \gamma \} \cap \mathscr{L}_1^{-1}(\mathcal{O}(\mathscr{G}))$.

Suppose $M \in \mathcal{O}(\mathscr{G}')$. Clearly, $M \in \Lambda_{\mathrm{lin}}(\Sigma_1')$, M is β-normal, and $\vdash_{\Sigma_1'} M : \gamma$. Let $P \in \mathcal{A}(\mathscr{G}')$ be such that $\mathscr{L}'(P) \twoheadrightarrow_{\beta} M$. Since $\vdash_{\Sigma_0'} P : s^{\gamma}$, we have $\vdash_{\Sigma_0} \mathscr{L}_0(P) : s$, so $|\mathscr{L}_0(P)|_{\beta} \in \mathcal{A}(\mathscr{G})$. Since $\mathscr{L}(\mathscr{L}_0(P)) = \mathscr{L}_1(\mathscr{L}'(P))$, we have $\mathscr{L}_1(M) = \mathscr{L}_1(|\mathscr{L}'(P)|_{\beta}) = |\mathscr{L}_1(\mathscr{L}'(P))|_{\beta} = |\mathscr{L}(\mathscr{L}_0(P))|_{\beta} = |\mathscr{L}(|\mathscr{L}_0(P)|_{\beta})|_{\beta} \in \mathcal{O}(\mathscr{G})$.

Claim. $\{ M \in \Lambda_{\mathrm{lin}}(\Sigma_1') \mid M \textit{ is } \beta\textit{-normal and } \vdash_{\Sigma_1'} M : \gamma \} \cap \mathscr{L}_1^{-1}(\mathcal{O}(\mathscr{G})) \subseteq \mathcal{O}(\mathscr{G}')$.

Suppose that $M \in \Lambda_{\mathrm{lin}}(\Sigma_1')$, M is β-normal, $\vdash_{\Sigma_1'} M{:}\gamma$, and $\mathscr{L}_1(M) \in \mathcal{O}(\mathscr{G})$. Let $\hat{M}[x_1,\ldots,x_n]$ be a constant-free linear λ-term such that $\hat{M}[a_1',\ldots,a_n'] = M$, where $a_1',\ldots,a_n' \in C_1'$. For $i = 1,\ldots,n$, let $\mathscr{L}_1(a_i') = a_i$. Then $\mathscr{L}_1(M) = \hat{M}[a_1,\ldots,a_n]$. Since $\mathscr{L}_1(M) \in \mathcal{O}(\mathscr{G})$, there is a $P \in \mathcal{A}(\mathscr{G})$ such that $\mathscr{L}(P) \twoheadrightarrow_{\beta} \hat{M}[a_1,\ldots,a_n]$. Let $\hat{P}[y_1,\ldots,y_m]$ be a constant-free linear λ-term such that

$\hat{P}[c_1, \ldots, c_m] = P$, where $c_1, \ldots, c_m \in C_0$. We have

$$y_1 : \tau_0(c_1), \ldots, y_m : \tau_0(c_m) \vdash \hat{P}[y_1, \ldots, y_m] : s. \tag{1}$$

Since $\mathscr{L}(P) = \hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] \twoheadrightarrow_\beta \hat{M}[a_1, \ldots, a_n]$ by non-erasing non-duplicating β-reduction, we can find, for $i = 1, \ldots, m$, a constant-free linear λ-term $\hat{N}_i$ with $\mathrm{FV}(\hat{N}_i) \subseteq \{x_1, \ldots, x_n\}$ such that

$$\hat{N}_i[x_1 := a_1, \ldots, x_n := a_n] = \mathscr{L}(c_i), \quad \hat{P}[\hat{N}_1, \ldots, \hat{N}_m] \twoheadrightarrow_\beta \hat{M}[x_1, \ldots, x_n].$$

For $1 \leq i \leq m$, let $N_i = \hat{N}_i[x_1 := a'_1, \ldots, x_n := a'_n]$, so that

$$\mathscr{L}_1(N_i) = \mathscr{L}(c_i). \tag{2}$$

Then

$$\hat{P}[N_1, \ldots, N_m] \twoheadrightarrow_\beta M \tag{3}$$

by non-erasing non-duplicating β-reduction. Since $\vdash_{\Sigma'_1} M : \gamma$, we get

$$\vdash_{\Sigma'_1} \hat{P}[N_1, \ldots, N_m] : \gamma \tag{4}$$

by the Subject Expansion Theorem.

Let $\mathscr{D}$ be the unique $\lambda{\to}_{\Sigma'_1}$-deduction of (4). For each $i = 1, \ldots, m$, $\mathscr{D}$ contains a subdeduction $\mathscr{D}_i$ of

$$\vdash_{\Sigma'_1} N_i : \beta_i \tag{5}$$

for some $\beta_i \in \mathscr{T}(A'_1)$, such that

$$y_1 : \beta_1, \ldots, y_m : \beta_m \vdash \hat{P}[y_1, \ldots, y_m] : \gamma. \tag{6}$$

It is easy to see that applying the lexicon $\mathscr{L}_1$ to each step of $\mathscr{D}$ gives a $\lambda{\to}_{\Sigma_1}$-deduction $\mathscr{D}'$ of

$$\vdash_{\Sigma_1} \hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] : \mathscr{L}(s).$$

Since $\hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] = \mathscr{L}(P)$, we see that $\mathscr{L}_1$ maps $\mathscr{D}_i$ to the unique $\lambda{\to}_{\Sigma_1}$-deduction of

$$\vdash_{\Sigma_1} \mathscr{L}(c_i) : \mathscr{L}(\tau(c_i)).$$

It follows that

$$\mathscr{L}_1(\beta_i) = \mathscr{L}(\tau(c_i)). \tag{7}$$

By (2), (5), and (7),

$$d_{\langle c_i, N_i, \beta_i \rangle} \in C'_0.$$

Let $\tau'_0(d_{\langle c_i, N_i, \beta_i \rangle}) = \delta_i$ for $i = 1, \ldots, m$. By the definition of τ'_0,

$$\langle \delta_1, \ldots, \delta_m, s^\gamma \rangle$$

is a most specific common anti-instance of

$$\langle \tau(c_1), \ldots, \tau(c_m), s \rangle \quad \text{and} \quad \langle \beta_1, \ldots, \beta_m, \gamma \rangle.$$

By the Principal Pair Theorem, it follows from (1) and (6) that

$$y_1 : \delta_1, \ldots, y_m : \delta_m \vdash \hat{P}[y_1, \ldots, y_m] : s^\gamma$$

and hence

$$\vdash_{\Sigma'_0} \hat{P}[d_{\langle c_1, N_1, \beta_1 \rangle}, \ldots, d_{\langle c_m, N_m, \beta_m \rangle}] : s^\gamma.$$

Therefore, $\hat{P}[d_{\langle c_1, N_1, \beta_1 \rangle}, \ldots, d_{\langle c_m, N_m, \beta_m \rangle}] \in \mathcal{A}(\mathscr{G}')$. Since

$$\begin{aligned} \mathscr{L}'(\hat{P}[d_{\langle c_1, N_1, \beta_1 \rangle}, \ldots, d_{\langle c_m, N_m, \beta_m \rangle}]) &= \hat{P}[\mathscr{L}'(d_{\langle c_1, N_1, \beta_1 \rangle}), \ldots, \mathscr{L}'(d_{\langle c_m, N_m, \beta_m \rangle})] \\ &= \hat{P}[N_1, \ldots, N_m], \end{aligned}$$

we conclude from (3) that $M \in \mathcal{O}(\mathscr{G}')$. □

4 String languages of ACGs

4.1 ACGs and full AFLs

Let us consider the family $\mathbf{L}_{\mathrm{str}}(m, n)$ of string languages generated by ACGs in $\mathbf{G}(m, n)$ $(m, n \geq 2)$. Lemma 3.2 implies that $\mathbf{L}_{\mathrm{str}}(m, n)$ is closed under substitution. Since $\mathbf{L}_{\mathrm{str}}(2, 2)$ includes all regular sets, it follows that $\mathbf{L}_{\mathrm{str}}(m, n)$ is closed under union, concatenation, Kleene star, and homomorphism.

We now prove that $\mathbf{L}_{\mathrm{str}}(m, n)$ is closed under intersection with regular sets.

Lemma 4.1 (Closure under intersection with regular sets) *Let $\mathscr{G} = \langle \Sigma_0, \Sigma_V, \mathscr{L}, s \rangle$ be a string ACG in $\mathbf{G}(m, n)$ and let R be a regular language over V. Then one can find an ACG $\mathscr{G}_{\cap R}$ in $\mathbf{G}(m, n)$ such that $\mathcal{O}(\mathscr{G}_{\cap R}) = \mathcal{O}(\mathscr{G}) \cap \{\, /w/ \mid w \in R \,\}$.*

Proof. By closure under union, it suffices to consider the case where R is an ϵ-free regular language over V. Then R is accepted by a nondeterministic finite automaton $\mathbf{M} = \langle Q, V, \delta, q_I, \{q_F\} \rangle$ without ϵ-transitions which has just one final state. Define a signature $\Sigma_{\mathbf{M}} = \langle Q, C_{\mathbf{M}}, \tau_{\mathbf{M}} \rangle$ by

$$C_{\mathbf{M}} = \{\, a^{r \to q} \mid a \in V, r \in \delta(q, a) \,\}, \qquad \tau_{\mathbf{M}}(a^{r \to q}) = r \to q.$$

Define a lexicon $\mathscr{L}_{\mathbf{M}} = \langle \sigma_{\mathbf{M}}, \theta_{\mathbf{M}} \rangle$ from $\Sigma_{\mathbf{M}}$ to Σ_V by

$$\sigma_{\mathbf{M}}(q) = o \quad \text{for all } q \in Q, \qquad \theta_{\mathbf{M}}(a^{r \to q}) = a.$$

Then $\mathscr{L}_{\mathbf{M}}$ is a relabeling, and we have $\vdash_{\Sigma_{\mathbf{M}}} N : q_F \to q_I$ if and only if $\mathscr{L}_{\mathbf{M}}(N) =_{\beta\eta} /w/$ for some $w \in R$. By Lemma 3.3, we can form an ACG $\mathscr{G}' = \langle \Sigma'_0, \Sigma_{\mathbf{M}}, \mathscr{L}', s^{q_F \to q_I} \rangle \in \mathbf{G}(m, n)$ such that

$$\mathcal{O}(\mathscr{G}') = \{\, M \in \Lambda_{\mathrm{lin}}(\Sigma_{\mathbf{M}}) \mid M \text{ is } \beta\text{-normal and } \vdash_{\Sigma_{\mathbf{M}}} M : q_F \to q_I \,\} \cap \mathscr{L}_{\mathbf{M}}^{-1}(\mathcal{O}(\mathscr{G})).$$

Then $\mathscr{G}_{\cap R} = \langle \Sigma'_0, \Sigma_V, \mathscr{L}_{\mathbf{M}} \circ \mathscr{L}', s^{q_F \to q_I} \rangle$ is the required ACG. □

To see that $\mathbf{L}_{\text{str}}(m,n)$ is closed under inverse homomorphism, we can use the following fact:

Fact ([7]). *If a family of languages includes the regular sets and is closed under substitution and intersection with regular sets, then it is closed under inverse homomorphism.*

Theorem 4.2 *The family of string languages generated by ACGs in* $\mathbf{G}(m,n)$ *is a substitution-closed full AFL for all* $m,n \geq 2$.

Corollary 4.3 *The family of string languages generated by ACGs is a substitution-closed full AFL.*

4.2 Lexicalized ACGs

Now let us consider the family $\mathbf{L}^{\text{lex}}_{\text{str}}(m,n)$ of string languages generated by ACGs in $\mathbf{G}(m,n) \cap \mathbf{Lex}$ $(m,n \geq 2)$. By Lemma 3.2, $\mathbf{L}^{\text{lex}}_{\text{str}}(m,n)$ is closed under substitution. Since $\mathbf{L}^{\text{lex}}_{\text{str}}(2,2)$ includes all ϵ-free regular sets, it follows that $\mathbf{L}^{\text{lex}}_{\text{str}}(m,n)$ is closed under union, concatenation, Kleene plus, and ϵ-free homomorphism. Lemma 4.1 holds with $\mathbf{G}(m,n) \cap \mathbf{Lex}$ in place of $\mathbf{G}(m,n)$, so $\mathbf{L}^{\text{lex}}_{\text{str}}(m,n)$ is also closed under intersection with regular sets.

We do not know whether $\mathbf{L}^{\text{lex}}_{\text{str}}(m,n)$ is closed under inverse homomorphism; this cannot be proved in the same way as for $\mathbf{L}_{\text{str}}(m,n)$. We can prove, however, that the family of string languages generated by all lexicalized ACGs is closed under inverse homomorphism. A homomorphism h from V_1^* to V_2^* is a *k-limited erasing* on $L \subseteq V_1^*$ if for every $uvw \in L$, $h(v) = \epsilon$ implies $|v| \leq k$.

Fact ([7]). *If a family of ϵ-free languages includes the ϵ-free regular sets and is closed under substitution, intersection with regular sets, and k-limited erasing for all k, then it is closed under inverse homomorphism.*

Lemma 4.4 *The family of string languages generated by lexicalized ACGs is closed under k-limited erasing for all k.*

Proof. Let $\mathscr{G} = \langle \Sigma, \Sigma_{V_1}, \mathscr{L}, s \rangle$ be a lexicalized string ACG generating $L \subseteq V_1^+$ and $h\colon V_1^* \to V_2^*$ be a k-limited erasing on L. We can assume without loss of generality that for every $c \in C$, $\mathscr{L}(c)$ contains exactly one occurrence of a constant. Let $\mathscr{L}_h$ be the first-order lexicon from Σ_{V_1} to Σ_{V_2} such that $\mathscr{L}_h(a) = /h(a)/$ for all $a \in V_1$. We construct a lexicalized ACG $\mathscr{G}' = \langle \Sigma', \Sigma_{V_2}, \mathscr{L}', s \rangle$ generating $h(L)$ as follows. We let $A' = A$ and $\mathscr{L}'(p) = \mathscr{L}(p)$ for all $p \in A'$. For each non-empty subset $\{c_1, \ldots, c_i\}$ of C with $i \leq k+1$ elements, C' contains the constant $d_{\{c_1,\ldots,c_i\}}$ with $\tau'(d_{\{c_1,\ldots,c_i\}}) = (\tau(c_1) \to \cdots \to \tau(c_i) \to s) \to s$ and $\mathscr{L}'(d_{\{c_1,\ldots,c_i\}}) = \mathscr{L}_h(\lambda x.x\mathscr{L}(c_1)\ldots\mathscr{L}(c_i))$, provided that this definition makes $d_{\{c_1,\ldots,c_i\}}$ lexical. We omit the proof of correctness. □

Theorem 4.5 *The family of string languages generated by lexicalized ACGs is a substitution-closed AFL.*

Since we know from [15] that $\bigcup_n \mathbf{L}^{\mathrm{lex}}_{\mathrm{str}}(2,n)$ is the set of ϵ-free languages in $\bigcup_n \mathbf{L}_{\mathrm{str}}(2,n)$, Theorem 4.2 implies the following:

Corollary 4.6 *The family of string languages generated by second-order lexicalized ACGs is a substitution-closed AFL.*

5 Tree languages of ACGs

Let us now look at the family $\mathbf{L}_{\mathrm{tree}}(m,n)$ of tree languages generated by ACGs in $\mathbf{G}(m,n)$ $(m \geq 2, n \geq 1)$.

The *tree concatenation* of $L_1, L_2 \subseteq \mathbb{T}_F$ is the result of tree substitution $L_1[c \leftarrow L_2]$ (where $c \in F_0$) and is denoted by $L_1 \cdot_c L_2$. The *iterated tree concatenation* $L^{*,c}$ is defined to be $\bigcup_{n\geq 0} L^{n,c}$, where

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

When variables are added to a ranked alphabet, they assume rank 0. A *tree homomorphism* is a mapping $h\colon F \to \mathbb{T}_{F' \cup X}$ such that $h(f) \in \mathbb{T}_{F' \cup \{x_1,\ldots,x_n\}}$ for all f in F_n. It is *linear* if x_i occurs at most once in $h(f)$, and *non-deleting* if for each $i = 1,\ldots,n$, x_i occurs at least once in $h(f)$. If h is a tree homomorphism and T is a tree, $h(T)$ is defined by induction as follows:

$$h(fT_1 \ldots T_n) = h(f)[h(T_1)/x_1, \ldots, h(T_n)/x_n].$$

The methods used in the previous section yield the following result about $\mathbf{L}_{\mathrm{tree}}(m,n)$:

Theorem 5.1 *The family of tree languages generated by ACGs in $\mathbf{G}(m,n)$ is closed under union, tree concatenation, linear non-deleting tree homomorphism, and intersection with regular tree languages.*

The closure under iterated concatenation requires a fresh method; we only have one that works with second-order ACGs.

Theorem 5.2 *The family of tree languages generated by ACGs in $\mathbf{G}(2,n)$ is closed under iterated tree concatenation.*

Given a symbol $c \notin F$, the *c-insertion* is a mapping $\langle c\rangle\colon \mathbb{T}_F \to \mathscr{P}(\mathbb{T}_{F\cup\{c\}})$ (c has rank 1 in $F \cup \{c\}$) defined as follows:

$$\langle c\rangle(fT_1 \ldots T_n) = \{\, c^k(fT'_1 \ldots T'_n) \mid k \geq 0 \text{ and } T'_i \in \langle c\rangle(T_i) \text{ for } i = 1,\ldots,n \,\},$$

where c^kT denotes the tree $\underbrace{c(\ldots(c}_{k \text{ times}}T)\ldots)$. This mapping is extended to a mapping from tree languages to tree languages by $\langle c\rangle(L) = \bigcup_{T\in L}\langle c\rangle(T)$.

Lemma 5.3 $\mathbf{L}_{\mathrm{tree}}(m,n)$ *is closed under c-insertion.*

We refer the reader to [3] for the definition of linear non-deleting (bottom-up or top-down) finite tree transducers (with ϵ-rules).

Theorem 5.4 *The family of tree languages generated by ACGs in* $\mathbf{G}(m,n)$ *is closed under linear non-deleting finite tree transduction.*

Proof. Let $L \subseteq \mathbb{T}_F$ be a tree language in $\mathbf{L}_{\text{tree}}(m,n)$, and let $c \notin F$. By Lemma 5.3, there is a tree ACG $\mathscr{G} = \langle \Sigma, \Sigma_{F\cup\{c\}}, \mathscr{L}, s\rangle$ in $\mathbf{G}(m,n)$ generating $\langle c\rangle(L)$.

Let $\mathbf{M} = \langle Q, F, F', \{q_F\}, \delta\rangle$ be a linear non-deleting bottom-up tree transducer (with ϵ-rules) with just one final state. Define a signature $\Sigma_{\mathbf{M}} = \langle Q, C_{\mathbf{M}}, \tau_{\mathbf{M}}\rangle$ by

$$C_{\mathbf{M}} = \{\, f_U^{q_1\to\cdots\to q_n\to q} \mid f(q_1(x_1),\ldots,q_n(x_n)) \to q(U) \in \delta \,\} \cup \{\, c_U^{q\to q'} \mid q(x_1) \to q'(U) \in \delta \,\},$$
$$\tau_{\mathbf{M}}(f_U^{q_1\to\cdots\to q_n\to q}) = q_1 \to \cdots \to q_n \to q, \qquad \tau_{\mathbf{M}}(c_U^{q\to q'}) = q \to q'.$$

Define a relabeling $\mathscr{L}^{\text{in}}_{\mathbf{M}} = \langle \sigma_{\mathbf{M}}, \theta^{\text{in}}_{\mathbf{M}}\rangle$ from $\Sigma_{\mathbf{M}}$ to $\Sigma_{F\cup\{c\}}$ by

$$\sigma_{\mathbf{M}}(q) = o \quad \text{for all } q, \qquad \theta^{\text{in}}_{\mathbf{M}}(f_U^{q_1\to\cdots\to q_n\to q}) = f, \qquad \theta^{\text{in}}_{\mathbf{M}}(c_U^{q\to q'}) = c.$$

By Lemma 3.3, we can find an ACG $\mathscr{G}' \in \mathbf{G}(m,n)$ such that $\mathcal{O}(\mathscr{G}') = \{\, M \in \Lambda_{\text{lin}}(\Sigma_{\mathbf{M}}) \mid M \text{ is } \beta\text{-normal and } \vdash M : q_F \,\} \cap (\mathscr{L}^{\text{in}}_{\mathbf{M}})^{-1}(\langle c\rangle(L))$. Define a first-order lexicon $\mathscr{L}^{\text{out}}_{\mathbf{M}} = \langle \sigma_{\mathbf{M}}, \theta^{\text{out}}_{\mathbf{M}}\rangle$ from $\Sigma_{\mathbf{M}}$ to $\Sigma_{F'}$ by

$$\theta^{\text{out}}_{\mathbf{M}}(f_U^{q_1\to\cdots\to q_n\to q}) = \lambda x_1 \ldots x_n.U, \qquad \theta^{\text{out}}_{\mathbf{M}}(c_U^{q\to q'}) = \lambda x_1.U.$$

By composing $\mathscr{G}'$ with $\mathscr{L}^{\text{out}}_{\mathbf{M}}$ we can form a desired ACG in $\mathbf{G}(m,n)$ generating the image of L under $\mathbf{M}$. □

The above theorem does not imply closure under inverse tree homomorphism. Although $\mathbf{L}_{\text{tree}}(2,1)$ (the family of regular tree languages) is closed under inverse tree homomorphism, we have no good reason to expect that the same holds of $\mathbf{L}_{\text{tree}}(m,n)$ with other choices of m,n.[4] However, one can express an inverse *linear* tree homomorphism with the composition of a c-insertion, an inverse relabeling, and a *third-order* lexicon, and this leads to the following result:

Theorem 5.5 *The family of tree languages generated by m-th order ACGs is closed under inverse linear tree homomorphism.*

References

[1] Aho, Alfred V. Indexed grammars—An extension of context-free grammars. *Journal of the Association for Computing Machinery* **15**, 647–671, 1968.

[4] Note that the family of tree languages generated by context-free tree grammars is not closed under h^{-1} even when h is linear and non-deleting ([2]).

[2] Arnold, André and Max Dauchet. Forêts algébriques et homomorphismes inverses. *Information and Control* **37**, 182–196, 1978.

[3] Comon, Hubert, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. Available online at http://www.grappa.univ-lille3.fr/tata/, 2002.

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

[5] Gécseg, Ferenc and Magnus Steinby. Tree languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Vol. 3: Beyond Words*, pages 1–68. Berlin: Springer, 1997.

[6] Ginsburg, Seymour and Sheila Greibach. Abstract families of languages. In *Studies in Abstract Families of Languages*, pages 1–32. Memoir No. 87. Providence, R.I.: American Mathematical Society, 1969.

[7] Greibach, Sheila and John Hopcroft. Independence of AFL operators. In *Studies in Abstract Families of Languages*, pages 33–40. Memoir No. 87. Providence, R.I.: American Mathematical Society, 1969.

[8] de Groote, Philippe. 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, 2001.

[9] de Groote, Philippe. Tree-adjoining grammars as abstract categorial grammars. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pages 101–106. Universitá di Venezial, 2002.

[10] de Groote, Philippe, Bruno Guillaume, and Sylvain Salvati. Vector addition tree automata. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 64–73, 2004.

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

[12] Hindley, J. Roger. BCK-combinators and linear λ-terms have types. *Theoretical Computer Science* **64**, 97–105, 1989.

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

[14] Joshi, Aravind K. and Yves Schabes. Tree-adjoining grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Vol. 3: Beyond Words*, pages 69–123. Berlin: Springer, 1997.

[15] Kanazawa, Makoto and Ryo Yoshinaka. Lexicalization of second-order ACGs. NII Technical Report. NII-2005-012E. National Institute of Informatics, Tokyo, 2005.

[16] Salvati, Sylvain. *Problèmes de Filtrage et Problèmes d'analyse pour les Grammaires Catégorielles Abstraites*. Doctoral thesis. Institut National Polytechnique de Lorraine, 2005.

[17] Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context-free grammars. *Theoretical Computer Science* **88**, 191–229, 1991.

[18] Weir, David J. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. dissertation. University of Pennsylvania, 1988.

[19] Yoshinaka, Ryo and Makoto Kanazawa. The complexity and generative capacity of lexicalized abstract categorial grammars. In Philippe Blache, Edward Stabler, Joan Busquets, and Richard Moot, editors, *Logical Aspects of Computational Linguistics: 5th International Conference, LACL 2005*, pages 330–346. Berlin: Springer, 2005.