Extensions and Restrictions of Abstract Categorial Grammars

（抽象的範疇文法の拡張と制限）

吉 仲　亮

Extensions and Restrictions of Abstract Categorial Grammars

Ryo Yoshinaka

Thesis submitted for the degree of
Doctor of Philosophy (Information Studies)
to the Graduate School of Interdisciplinary Information Studies,
the University of Tokyo

September 2006

Thesis Committee: Jun'ichi Tsujii (Chair)
Makoto Kanazawa
Hiroshi Nakagawa
Ryu Hasegawa
Kumiko Tanaka-Ishii

Acknowledgment

I am very much grateful to Dr. Makoto Kanazawa, who led and encouraged me to get motivated to work on Abstract Categorial Grammars, the subject of this thesis, and gave me crucial advice and comments. He has supported me throughout my dissertation research, not only while he was my official supervisor, but also after he moved from the University of Tokyo to the National Institute of Informatics.

Dr. Kanazawa organized the Workshop on Lambda Calculus and Formal Grammar twice in Tokyo. I got explicit and implicit suggestions for my research from the talks and discussions by speakers at the workshop, Dr. Philippe de Groote, who is the advocator of Abstract Categorial Grammars, Dr. Reinhard Muskens, Dr. Sylvain Pogodalla, Dr. Kazushige Terui, Dr. Sylvain Salvati and Dr. Kanazawa. It is my pleasure to express my gratitude to them.

Dr. de Groote and Dr. Pogodalla offered me kind hospitality and valuable academic experience when I visited their laboratory. I deeply appreciate their support.

Dr. Salvati provided me many occasions to discuss Abstract Categorial Grammars and the Lambda Calculus with him and gave me many interesting suggestions and useful advice. I thank him very much for his help and friendship.

I would like to thank Dr. Makoto Tatsuta and the members of his seminar, Dr. Kanazawa, Dr. Terui, Mr. Daisuke Kimura and Mr. Takayuki Koai. I talked about my work in progress there and got many helpful comments from them.

Last but not least, I am sincerely indebted to the members of the thesis committee, Dr. Jun'ichi Tsujii, my official supervisor, Dr. Hiroshi Nakagawa, my official vice-supervisor, Dr. Ryu Hasegawa, Dr. Kumiko Tanaka-Ishii and Dr. Kanazawa. They gave me not only valuable criticism but also warm encouragement.

Abstract

Phillippe de Groote has introduced a new grammar formalism called *Abstract Categorial Grammars (ACGs)* based on simply typed linear lambda calculus. Not only can ACGs explain various problematic linguistic phenomena in elegant ways, but also they encode mildly context-sensitive formalisms in straightforward ways. This thesis is concerned with the mathematical properties of some simple extensions and restrictions of ACGs. In particular, the aspect of ACGs as a generalization of well-established grammar formalisms including those mildly context-sensitive formalisms is investigated. Extensions and restrictions of ACGs that we discuss are non-linear ACGs, lexicalized forms of ACGs, and two-dimensional ACGs.

While the linearity constraint on ACGs is considered reasonable for several reasons, non-linear λ-terms enable ACGs to represent some linguistic phenomena in a more natural fashion. We show that relaxing the linearity constraint by allowing vacuous λ-abstraction does not increase the expressive power of ACGs. Our conversion entails that allowing or disallowing deleting rules in some existing grammar formalisms does not change their generative power.

Although the origin of ACGs is located in a history of categorial grammars, ACGs are not necessarily lexicalized grammars by definition, unlike usual categorial grammar formalisms. Not only from the lexicalist's point of view but also from a computational point of view, the restriction to lexicalized ACGs is an important issue. We show that some subclasses of ACGs admit lexicalization in the sense that each grammar in one of these subclasses can be converted into an equivalent lexicalized grammar belonging to the same class.

Two-dimensional extensions of ACGs define relations among languages generated by ACGs, and are able to model the relations between sentences and their meanings of natural language. We investigate the generative capacity of two-dimensional ACGs through encoding some well-known tree transducers.

Contents

Chapter 1

Introduction

One important attractive feature of so-called *categorial grammars* as a tool for modeling natural language is that a sentence and its meaning are derived from one common proof in the underlying logic. However, several linguistic phenomena are found that cannot be explained by the Lambek Grammar [29], which is the most representative categorial grammar formalism. Among several modifications (e.g., Morrill [37], Moortgat [35,36], Oehrle [40,41]) of the Lambek Grammar, de Groote [15] has introduced a new grammar formalism called *Abstract Categorial Grammars (ACGs)*, which is based on the implicational fragment of the *linear logic*. De Groote's approach is essentially the same as Oehrle's, but the ACG formalism radically simplifies Oehrle's formalism. While in Oehrle's formalism, the underlying logic contains the conjunction and word forms are represented by a special kind of typed λ-terms, ACGs represent word forms and meanings as well as the derivation structures for them by simply typed linear λ-terms. While Muskens [38] has independently proposed a formalism almost equivalent to ACGs, we deal with ACGs because of their formal simplicity and tractability for mathematical discussion.

ACGs have a number of theoretically and linguistically attractive features as a grammar formalism for natural language. Although ACGs are not a variety of Lambek grammars, they inherit the virtues of categorial grammars. Moreover, ACGs give elegant explanations for several linguistic phenomena that cannot be treated well by Lambek grammars, such as the ambiguity of the scopes of quantifiers in a sentence like "`every man loves a woman`". While usual categorial grammars treat word forms and meanings asymmetrically, ACGs treat word forms as λ-terms as well as meanings. This simplification enables ACGs generate diverse types of languages, because typed λ-terms, which constitute the languages of ACGs, can represent diverse types of data including not only strings and meaning representations,

but also trees, lists, and so on. Besides, the simply typed linear lambda calculus has been studied very well so far. Hence we have plenty of useful tools for investigating the mathematical properties of ACGs.

The mathematical properties of ACGs have close relationship with so-called *mildly context-sensitive grammars* rather than usual categorial grammars. Since some linguistic phenomena, such as cross-serial dependencies in Dutch, were found that cannot be captured by context-free grammars (CFGs), several cautious extensions of CFGs, called mildly context-sensitive grammars, have been proposed and studied. Some formalisms are known to be equivalent; classes of languages defined by combinatory categorial grammars, head grammars, linear indexed grammars, and tree adjoining grammars (TAGs) are all equivalent [22, 55, 56]. Moreover, linear context-free rewriting systems (LCFRSs), multiple context-free grammars, minimalist grammars, and multi-component TAGs (MCTAGs) define the same class of languages [33, 34, 49, 56]. Recent studies [16, 18, 19, 58] on ACGs have shown that ACGs are powerful enough to encode those types of mildly context-sensitive grammars. They have presented straightforward encodings of CFGs, TAGs, non-deleting non-duplicating context-free tree grammars (CFTGs), LCFRSs, MCTAGs, where derivation structures in the original grammar are preserved. For instance, the ACG encoding a TAG generates derived trees through λ-terms encoding derivation trees of the TAG. Moreover, de Groote and Pogodalla [19] implicitly introduce a hierarchy of ACGs according to the complexity of λ-terms and types appearing in each ACG, and then associate the variety of context-free formalisms, namely, CFGs, non-deleting non-duplicating CFTGs, and LCFRSs, with the hierarchy of ACGs. In this sense, the ACG can be thought of as a generalization of mildly context-sensitive formalisms. To study ACGs would be to study those well-established grammar formalisms. In fact, de Groote and Pogodalla [19] state "Abstract Categorial Grammars ... should rather be seen as the kernel of a grammatical framework ... in which other existing grammatical models may be encoded".

This thesis is devoted to studying the mathematical properties of some extensions and restrictions of ACGs. In particular, the aspect of ACGs that generalizes well-established grammar formalisms is investigated. Through discussion on the three main issues of this thesis, *elimination of deleting operations*, *lexicalization*, and *two-dimensional formalisms*, it is demonstrated that ACGs can be thought of as a generalization of existing grammars in various senses.

Elimination of deleting operations.

Preceding research on several grammar formalisms have shown that deleting operations in a grammar can be eliminated preserving the language. We

show that a similar result holds for ACGs, which generalizes the results by preceding research on elimination of deleting operations in some grammar formalisms.

Lexicalization.

A grammar is called *lexicalized* if each of its lexical entries that contribute to deriving an element of the language contains an item that explicitly appears on the surface of the derived structure. We show that a certain subclass of ACGs admits lexicalization. As Schabes [47] has shown that every finitely ambiguous ACG can be lexicalized as a TAG, our lexicalization method converts the ACG encoding a CFG into the one encoding a lexicalized TAG as a string generator.

Two-dimensional formalisms.

Not a few grammar formalisms, which define languages, have been extended into *two-dimensional formalisms*, in which grammars define relations between two languages. We present some encoding methods for existing two-dimensional formalisms by two-dimensional ACGs.

Overview of This Thesis

In the next chapter, we give a formal definition of ACGs and see basic mathematical properties of ACGs as well as the results on ACGs obtained by preceding research. In particular, it is shown that if the universal membership problem or the non-emptiness problem of ACGs is decidable, then so is the multiplicative exponential fragment of the linear logic, which is in open, and that non-semilinear languages can be generated by ACGs. Those results might imply that ACGs have too much powerful generative capacity and intractable computational complexity for modeling natural language, so finding an appropriate restriction on ACGs is desired.

Chapter 3 is concerned with non-linear extensions of ACGs. While ACGs are based on the simply typed linear lambda calculus, 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, preceding research concerning mildly context-sensitive grammars has shown that deleting operations in grammars can be eliminated preserving their languages. Namely, Seki et al. [49] have shown that every multiple context-free grammar (MCFG) has an equivalent LCFRS, which can be thought of as an MCFG that has no deleting operations, and Fujiyoshi [10] has established the equivalence between non-duplicating monadic CFTGs and non-deleting non-duplicating monadic CFTGs. Along this line, we show that the generative capacity of

the usual ACGs is equivalent to that of the more liberal type of ACGs called *affine ACGs*, where vacuous λ-abstraction is allowed. Our conversion from an affine ACG into an equivalent linear ACG derives the results by Seki et al. and by Fujiyoshi as corollaries. This demonstrates that not only individual mildly context-sensitive grammars are encodable by ACGs, but also some transformations of grammars can be encoded in the ACG formalism.

In Chapter 4, we focus on *lexicalized ACGs*. A grammar is called lexicalized if each of its lexical entries that contribute to deriving an element of the language contains an item that explicitly appears on the surface of the derived structure. From the point of lexicalists' view, which thinks that linguistic phenomena should be accounted for by the inherent information in the lexical entries, to be lexicalized is a natural requirement. Schabes [47] has shown that every finitely ambiguous TAG admits lexicalization. Besides, because each production of a CFG is encoded as a lexical entry in the ACG encoding the CFG, a CFG in Greibach normal form can be considered lexicalized. While usual categorial grammars are lexicalized by definition, ACGs are not necessarily lexicalized. We first show that the universal membership problem for lexicalized ACG is NP-complete. This result contrasts with the one for general ACGs and suggests that the restriction to lexicalized ACGs may be preferable from the point of view of computational complexity as well as of lexicalism. Meanwhile, Salvati has introduced the notion of *semilexicalized ACGs* by relaxing the notion of lexicalized ACGs and shown that the languages generated by semilexicalized ACGs are in NP. His result suggests that the generative capacity of semilexicalized ACGs is not very much richer than that of lexicalized ACGs. The main result of Chapter 4 is lexicalization of semilexicalized ACGs. We show that each grammar in one level of the ACG hierarchy can be converted into an equivalent lexicalized grammar belonging to the same level if the level is higher than a certain level. Our lexicalization, however, generalizes neither lexicalization of TAGs nor Greibach normalization of CFGs, since the ACGs encoding TAGs or CFGs are located at a lower level than that level. Nevertheless, the fact that our lexicalization method converts ACGs encoding CFGs into lexicalized ACGs of the form encoding TAGs as *string generators* has a certain correspondence with Schabes's lexicalization method that lexicalizes finitely ambiguous CFGs as TAGs.

Chapter 5 is devoted to investigating the generative capacity of *two-dimensional ACGs*. Lambek grammars are two-dimensional grammars in the sense that they generate pairs of a sentence and its meaning. Although the ACG formalism is designed to be easily extended into a two-dimensional formalism, there is little research on the mathematical properties of two-dimensional ACGs. In order to evaluate the generative capacity of two-

dimensional ACGs, we present encoding methods that convert several existing two-dimensional formalisms, which define relations between two languages, into two-dimensional ACGs. In particular, a precise characterization of the class of non-deleting non-duplicating macro tree transducers [2,6] is given in the ACG hierarchy. Besides, it is shown that synchronous TAGs [50, 53] are encodable by two-dimensional ACGs. These results show that the ACG formalism generalizes existing two-dimensional formalisms as well as one-dimensional formalisms. On the other hand, we show that several properties of usual one-dimensional ACGs hold also for two-dimensional ones. In particular, as Salvati has shown that a part of the hierarchy of one-dimensional ACGs collapses, we show that the corresponding part of the hierarchy of two-dimensional ACGs also collapses through encoding deterministic tree-walking transducers [1] by two-dimensional ACGs.

The subject of Chapter 6 is a restricted kind of higher-order matching. The higher-order matching problem is an important topic in computer science. Parsing with ACGs has a close connection with a special form of the higher-order matching problem called *linear interpolation*. We show that linear interpolation is NP-complete. While each variable occurs exactly once in a linear λ-term, the number of occurrences of constants are not restricted. It is natural to ask whether the NP-hardness of linear interpolation still holds when we exclude multiple occurrences of constants from linear λ-terms. We answer the question in the affirmative.

Chapter 2

Abstract Categorial Grammars

2.1 Definitions and Notations

$\varnothing$ is the empty set, ε is the empty string, $\mathbb{Z}$ is the set of integers, $\mathbb{N}$ is the set of non-negative integers. $|X|$ denotes the cardinality of X if X is a set, and the length of X if X is a sequence. The sequence consisting of elements $X_1, \ldots, X_n$ in this order is denoted by $\langle X_1, \ldots, X_n \rangle$. If no confusion occurs, in particular if each X_i is not a sequence, we sometimes write $X_1 \ldots X_n$ for $\langle X_1, \ldots, X_n \rangle$. If $\vec{Y}$ denotes $\langle Y_1, \ldots, Y_m \rangle$, $X_1 \ldots X_m \vec{Y} Z_1 \ldots Z_l$ should be read as $\langle X_1, \ldots, X_n, Y_1, \ldots, Y_m, Z_1, \ldots, Z_l \rangle$ and so on. $X_1 \ldots X_m$ is said to be a *subsequence* of a sequence $\vec{Y}$ if there are some possibly empty sequences $\vec{Y}_0, \ldots, \vec{Y}_m$ such that $\vec{Y} = \vec{Y}_0 X_1 \vec{Y}_1 X_2 \ldots \vec{Y}_{m-1} X_m \vec{Y}_m$. For a fixed index set I and a set represented as $S = \{ X_i \mid i \in I \}$, we write $\langle X_i \in S \mid \phi(i) \rangle$ (or $\langle X_i \rangle_{\phi(i)}$) for an appropriate sequence of X_i such that $\phi(i)$ holds, if the order of the elements is determined uniquely from the context, or can be arbitrarily determined. Standard language-theoretic notions like *concatenation*, *Kleene star* $(*)$, *homomorphism* etc. are defined as usual.

2.1.1 Lambda-Terms

Types 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 $(\alpha \to \beta) \in \mathcal{T}(\mathscr{A})$.

For $\alpha \in \mathcal{T}(\mathscr{A})$, $|\alpha|$ represents the number of occurrences of atomic types in α. The set $\mathcal{ST}^+(\alpha)$ of *positive subtypes* of a type α, the set $\mathcal{ST}^-(\alpha)$ of *negative subtypes* of a type α, and the set $\mathcal{ST}(\alpha)$ of *subtypes* of a type α are defined as follows:

- $\mathcal{ST}^+(p) = \{p\}$, $\mathcal{ST}^-(p) = \varnothing$ for $p \in \mathscr{A}$.
- $\mathcal{ST}^+((\alpha \to \beta)) = \{(\alpha \to \beta)\} \cup \mathcal{ST}^+(\beta) \cup \mathcal{ST}^-(\alpha)$, and $\mathcal{ST}^-((\alpha \to \beta)) = \mathcal{ST}^-(\beta) \cup \mathcal{ST}^+(\alpha)$.
- $\mathcal{ST}(\alpha) = \mathcal{ST}^+(\alpha) \cup \mathcal{ST}^-(\alpha)$.

Let $\mathrm{ord} : \mathcal{T}(\mathscr{A}) \to \mathbb{N}$ be defined as

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

We say that a type α is *n-th order* if $\mathrm{ord}(\alpha) \leq n$. We use o, p, q, r, s for atomic types and early lower case Greek letters $\alpha, \beta, \gamma, \ldots$ for types unless otherwise noted. We write $\alpha_1 \to \cdots \to \alpha_n \to \beta$ for $(\alpha_1 \to (\cdots \to (\alpha_n \to \beta)\ldots))$ and if $\vec{\alpha} = \alpha_1 \ldots \alpha_n$, $\vec{\alpha} \to \beta$ means $\alpha_1 \to \cdots \to \alpha_n \to \beta$. $\alpha^n \to \beta$ stands for $\underbrace{\alpha \to \cdots \to \alpha}_{n\text{-times}} \to \beta$. A second-order type $\alpha \in \mathcal{T}(\mathscr{A})$ is said to be *n-ary* if $\alpha = p_1 \to \cdots \to p_n \to q$ for some $p_i, q \in \mathscr{A}$. Thus an atomic type is called a *nullary* second-order type.

Higher-Order Signatures 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 function from $\mathscr{C}$ to $\mathcal{T}(\mathscr{A})$ called a *type assignment*. We say that a higher-order signature Σ is *n-th order* if $\mathrm{ord}(\Sigma) \leq n$ where $\mathrm{ord}(\Sigma) = \max\{\, \mathrm{ord}(\tau(\mathsf{a})) \mid \mathsf{a} \in \mathscr{C} \,\}$. When we write Σ, Σ', Σ_i etc. to denote higher-order signatures, we assume $\Sigma = \langle \mathscr{A}, \mathscr{C}, \tau \rangle$, $\Sigma' = \langle \mathscr{A}', \mathscr{C}', \tau' \rangle$, $\Sigma_i = \langle \mathscr{A}_i, \mathscr{C}_i, \tau_i \rangle$ and so forth.

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

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

We assume that every occurrence of a variable x in one term has the same type; e.g., $\lambda x^p.x^{p\to q}x^p$ is disallowed. Constants and variables are called *atomic terms*, terms of the form (MN) are called *application terms*, terms of the form $(\lambda x^\alpha.M)$ are called *abstraction terms*. 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. We write $M_1 \dots M_n$ for $((\dots(M_1M_2)\dots)M_n)$, $\lambda x_1 \dots x_n.M$ for $(\lambda x_1.(\lambda x_2.\dots.(\lambda x_n.M)\dots))$, and so on. If $\vec{x} = x_1 \dots x_n$ and $\vec{M} = M_1 \dots M_m$, then $\lambda\vec{x}.M_0\vec{M}$ represents the term $\lambda x_1 \dots x_n.M_0M_1 \dots M_m$. The notions of free variables, closed terms, β-normal form, $\beta\eta$-normal form, η-long form, are defined as usual (see a standard text book, e.g. [20]). A term M is a *combinator* iff M is closed and M contains no constants. The *head* of M is the atomic term x if $M = \lambda\vec{x}.\mathsf{x}\vec{M}$. In a term $\mathsf{x}M_1 \dots M_m$, each occurrence of M_i is said an *argument* of the head occurrence of x, and conversely the *functor* of the occurrence of M_i is the head occurrence of x. We often say that *a term M is k-th order (k-ary)* if $\tau(M)$ is k-th order (k-ary).

A capture-avoiding substitution of a term M for a free variable x is denoted by $[M/x]$. When $\vec{x} = x_1 \dots x_n$ and $\vec{M} = M_1 \dots M_n$, $[\vec{M}/\vec{x}]$ means the simultaneous substitution $[M_1/x_1, \dots, M_n/x_n]$. $[M_i/x_i]_{\phi(i)}$ means the simultaneous substitution $[\langle M_i\rangle_{\phi(i)}/\langle x_i\rangle_{\phi(i)}]$. For a substitution σ, $N\sigma$ denotes the term obtained from N by applying the substitution and $\lambda xyz.LMN\sigma$ means $\lambda xyz.((LMN)\sigma)$. A substitution that may capture free variables is denoted by $[x := M]$. Thus, $\lambda x.y[x/y] = \lambda x.x$, $(\lambda x.y)[x/y] = \lambda z.x$, $(\lambda x.y)[y := x] = \lambda x.x$, and so on.

As usual, let $\twoheadrightarrow_\beta$, $=_\beta$, $=_{\beta\eta}$, $\equiv$ denote β-reduction, β-equality, $\beta\eta$-equality, and α-equality respectively. A *long normal term* means a η-long β-normal term. The $\beta\eta$-equality is often simply written by $=$. $|M|_\beta$, $|M|_{\beta\eta}$ and $\mathrm{Fv}(M)$ respectively represent the β-normal form, $\beta\eta$-normal form, and the set of free variables of M. We use upper case italic letters $M, N, P, \dots$ for terms, late lower case italic letters $x, y, z, \dots$ for variables, and sanserif $\mathsf{a}, \mathsf{A}, \dots$ for constants in principle.

The *size* of a term is defined by

$$\begin{aligned} \mathrm{size}(\mathsf{a}) = \mathrm{size}(x) &= 1, \\ \mathrm{size}(MN) &= \mathrm{size}(M) + \mathrm{size}(N), \\ \mathrm{size}(\lambda x.M) &= \mathrm{size}(M) + 1. \end{aligned}$$

For a term $M \in \Lambda(\Sigma)$ and $\mathsf{a} \subseteq \mathscr{C}$, $\#_{\mathsf{a}}(M)$ denotes the number of occurrences of the constant a in M. Moreover, for $\mathscr{C}' \subseteq \mathscr{C}$, $\#_{\mathscr{C}'}(M)$ denotes the total number of occurrences of the constants in $\mathscr{C}'$. This notation is defined in the same way for strings.

A term is said to be *linear* if every λ-abstraction binds exactly one occurrence of a variable and no free variable of it occurs more than once.

Convention 2.1. Since this thesis mainly concerns linear λ-terms, we denote the set of linear λ-terms on Σ by $\Lambda(\Sigma)$ and moreover we omit the modifier "linear", so if we say simply a term, it means a linear λ-term hereafter.

Strings, Trees and Lambda-Terms A second-order signature Σ is called *string signature* if $\mathscr{A} = \{o\}$ and $\tau(\mathsf{a}) = o \to o$ for every $\mathsf{a} \in \mathscr{C}$. We write *str* for $o \to o$. For an (unranked) alphabet V, Σ_V denotes the corresponding string signature such that $\mathscr{C}_V = V$. For a string $\mathsf{a}_1 \dots \mathsf{a}_n \in V^*$, we define the corresponding closed linear term as

$$/\mathsf{a}_1 \dots \mathsf{a}_n/ \equiv \lambda z^o \mathsf{a}_1(\mathsf{a}_2(\dots(\mathsf{a}_n z)\dots)).$$

The concatenation is represented by the combinator $\mathbf{B} \equiv \lambda x^{str} y^{str} z^o . x(yz)$. For $\mathsf{w}_1, \mathsf{w}_2 \in V^*$, we have

$$\mathbf{B}/\mathsf{w}_1//\mathsf{w}_2/ = /\mathsf{w}_1\mathsf{w}_2/.$$

For notational convenience, we write $M{+}N$ instead of $\mathbf{B}MN$ for two terms of type *str*. Moreover, by the associativity of $+$, i.e., $(L{+}M){+}N = L{+}(M{+}N)$, we can omit parentheses as $L{+}M{+}N$ if $\beta\eta$-equivalent terms can be identified.

A *ranked alphabet* is a pair $\langle F, \rho \rangle$ where F is an (unranked) alphabet and ρ is a function from F to the non-negative integers. $F^{(n)}$ denotes the set $\{\, \mathsf{f} \in F \mid \rho(\mathsf{f}) = n \,\}$. The set $\mathbb{T}(\langle F, \rho \rangle)$ of *trees* over $\langle F, \rho \rangle$ is the smallest set such that

- if $\mathsf{f} \in F^{(n)}$ and $M_1, \dots, M_n \in \mathbb{T}(\langle F, \rho \rangle)$, then $(\mathsf{f} M_1 \dots M_n) \in \mathbb{T}(\langle F, \rho \rangle)$.

The outer most pair of parentheses of a tree is usually omitted. For a possibly infinite set $\mathscr{X}$ of variables, by $\mathbb{T}(\langle F, \rho \rangle \cup \mathscr{X})$ we denote the set of trees that may contain variables as symbols of rank 0. A second-order signature Σ is called a *tree signature* if $\mathscr{A} = \{o\}$. For a ranked alphabet $\langle F, \rho \rangle$, we define the corresponding tree signature as $\Sigma_{\langle F,\rho \rangle} = \langle \{o\}, F, \{\, \mathsf{f} \mapsto o^{\rho(\mathsf{f})} \to o \mid \mathsf{f} \in F \,\} \rangle$. We often identify a ranked alphabet $\langle F, \rho \rangle$ with the corresponding tree signature $\Sigma_{\langle F,\rho \rangle}$. Then $\mathbb{T}(\Sigma)$ is the subset of $\Lambda(\Sigma)$ whose elements are variable-free and of atomic types. The *yield string* $\mathrm{yield}(M)$ of a tree M is defined as

$$\begin{aligned} \mathrm{yield}(\mathsf{f}) &= \mathsf{f} && \text{if } \mathsf{f} \in F^{(0)}, \\ \mathrm{yield}(\mathsf{f} M_1 \dots M_n) &= \mathrm{yield}(M_1) \dots \mathrm{yield}(M_n) && \text{if } \mathsf{f} \in F^{(n)} \text{ with } n \geq 1. \end{aligned}$$

2.1.2 Abstract Categorial Grammars

Definition 2.2 (Lexicon). 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. On a lexicon $\mathscr{L} = \langle \sigma, \theta \rangle$, the function $\widehat{\mathscr{L}}$ from $\Lambda(\Sigma_0)$ to $\Lambda(\Sigma_1)$ is uniquely determined as follows:

$$\begin{aligned}
\widehat{\mathscr{L}}(x^\alpha) &= x^{\sigma(\alpha)} \quad \text{for } \alpha \in \mathcal{T}(\mathscr{A}_0), \\
\widehat{\mathscr{L}}(\mathsf{a}) &= \theta(\mathsf{a}) \quad \text{for } \mathsf{a} \in \mathscr{C}_0, \\
\widehat{\mathscr{L}}(\lambda x^\alpha.M) &= \lambda x^{\sigma(\alpha)}.\widehat{\mathscr{L}}(M) \quad \text{for } \lambda x^\alpha.M \in \Lambda(\Sigma_0), \\
\widehat{\mathscr{L}}(MN) &= \widehat{\mathscr{L}}(M)\widehat{\mathscr{L}}(N) \quad \text{for } MN \in \Lambda(\Sigma_0).
\end{aligned}$$

Indeed, $\widehat{\mathscr{L}}(M)$ always a well-typed λ-term if so is M; if M has type α, then $\widehat{\mathscr{L}}(M)$ has type $\sigma(\alpha)$. Hereafter we simply write $\mathscr{L}$ for σ or $\widehat{\mathscr{L}}$ depending on the context. A lexicon $\mathscr{L} = \langle \sigma, \theta \rangle$ is *n-th order* if $\mathrm{ord}(\mathscr{L}) = \max\{\, \mathrm{ord}(\sigma(p)) \mid p \in \mathscr{A}_0 \,\} \leq n$. For two lexicons $\mathscr{L}_1 = \langle \sigma_1, \theta_1 \rangle : \Sigma_0 \to \Sigma_1$ and $\mathscr{L}_2 = \langle \sigma_2, \theta_2 \rangle : \Sigma_1 \to \Sigma_2$, it is easily seen that $\sigma_2 \circ \sigma_1$ and $\theta_2 \circ \theta_1$ are also compatible. Thus, the composite lexicon $\mathscr{L}_2 \circ \mathscr{L}_1 : \Sigma_0 \to \Sigma_2$ can be defined as $\mathscr{L}_2 = \langle \sigma_2 \circ \sigma_1, \theta_2 \circ \theta_1 \rangle$, where it holds that $\mathrm{ord}(\mathscr{L}_2 \circ \mathscr{L}_1) \leq \mathrm{ord}(\mathscr{L}_1) + \mathrm{ord}(\mathscr{L}_2) - 1$.

Definition 2.3 (de Groote [15]). 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 lexicon from Σ_0 to Σ_1,
- $s \in \mathscr{A}_0$ is called the *distinguished type*.

When we use the modifier *abstract* or *object*, it specifies the vocabulary that a given type or term belongs to. Thus, we speak of *abstract atomic types*,

object atomic types, *abstract constants*, *object constants*, etc. 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 2.4. 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 follows:

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

The abstract language can be thought 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. Two ACGs are *equivalent* iff their object languages coincide. For conciseness we write $M \in \mathcal{A}(\mathscr{G})$ ($P \in \mathcal{O}(\mathscr{G})$) instead of $|M|_{\beta\eta} \in \mathcal{A}(\mathscr{G})$ ($|P|_{\beta\eta} \in \mathcal{O}(\mathscr{G})$) even if M (P) is not $\beta\eta$-normal.

A *string ACG* is an ACG whose object vocabulary is a string signature and whose distinguished type is mapped to the type *str*. A *tree ACG* is an ACG whose object vocabulary is a tree signature and whose distinguished type is mapped to the type o. By $\mathbf{G}_{\text{string}}$ and $\mathbf{G}_{\text{tree}}$, we denote the classes of string ACGs and tree ACGs, respectively.

Let us denote by $\mathbf{G}(m, n)$ the subclass of ACGs $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s\rangle$ such that $\mathrm{ord}(\Sigma_0) \leq m$ and $\mathrm{ord}(\mathscr{L}) \leq n$. An ACG in $\mathbf{G}(m, n)$ is also called an *m-th order ACG*. For a second-order ACG, its lexicon is said to be *semi-relabeling* if it is first-order and it maps each abstract constant to an object constant or the identity $\lambda z^o.z$. A *relabeling* lexicon is a semi-relabeling lexicon which maps every abstract constant to an object constant. $\mathbf{G}(2, 1(\mathrm{sr}))$ and $\mathbf{G}(2, 1(\mathrm{r}))$ denote the class of second-order ACGs whose lexicons are semi-relabeling and relabeling, respectively. A second-order lexicon $\mathscr{L}$ is said to be *monadic* if for each abstract atomic type $p \in \mathscr{A}_0$, $\mathscr{L}(p)$ is either an atomic or a unary type. $\mathbf{G}(2, 2(\mathrm{mon}))$ is the class of second-order ACGs whose lexicons are monadic.

We specify a subclass of ACGs by combining these notations; for instance, $\mathbf{G}_{\text{tree}}(2, 1(\mathrm{r}))$ denotes the class of tree ACGs whose abstract vocabulary is second-order and lexicon is relabeling.

Example 2.5. Let $\mathscr{G}_1 = \langle \Sigma_0, \Sigma_1, \mathscr{L}_1, s\rangle \in \mathbf{G}_{\text{string}}$ consist of the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}_1(\mathsf{x})$
M	n	$/\mathtt{man}/$
W	n	$/\mathtt{woman}/$
L	$np^2 \to s$	$\lambda yx.x + /\mathtt{loves}/ + y$
A	$n \to (np \to s) \to s$	$\lambda yw.w(/\mathtt{a}/ + y)$
E	$n \to (np \to s) \to s$	$\lambda xw.w(/\mathtt{every}/ + x)$

where $\mathscr{L}(s) = \mathscr{L}(np) = \mathscr{L}(n) = \mathit{str}$.

Moreover, let $\mathscr{G}_2 = \langle \Sigma_0, \Sigma_2, \mathscr{L}_2, s \rangle$ consist of the following lexical entries:[1]

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}_2(\mathsf{x})$
M	n	$\lambda z.\mathbf{man}\, z$
W	n	$\lambda z.\mathbf{woman}\, z$
L	$np^2 \to s$	$\lambda yx.\mathbf{love}\, y\, x$
A	$n \to (np \to s) \to s$	$\lambda w_1 w_2.\exists y((w_1 y) \wedge (w_2 y))$
E	$n \to (np \to s) \to s$	$\lambda w_1 w_2.\forall x((w_1 x) \longrightarrow (w_2 x))$

where $\mathscr{L}(s) = t$, $\mathscr{L}(np) = e$, $\mathscr{L}(n) = e \to t$. Let

$$M = \mathsf{AW}(\lambda x^{np}.\mathsf{EM}(\lambda y^{np}.\mathsf{L}xy)),$$
$$N = \mathsf{EM}(\lambda y^{np}.\mathsf{AW}(\lambda x^{np}.\mathsf{L}xy)).$$

We have

$$\mathscr{L}_1(M) = \mathscr{L}_1(N) = /\mathtt{every\ man\ loves\ a\ woman}/$$
$$\mathscr{L}_2(M) = \exists y((\mathbf{woman}\, y) \wedge (\forall x((\mathbf{man}\, x) \longrightarrow (\mathbf{love}\, x\, y))))$$
$$\mathscr{L}_2(N) = \forall x((\mathbf{man}\, x) \longrightarrow (\exists y((\mathbf{woman}\, y) \wedge (\mathbf{love}\, x\, y)))).$$

This way the above two ACGs give an explanation of the ambiguity of the scopes of the quantifiers.

2.2 Basic Properties of Abstract Categorial Grammars

2.2.1 Hierarchy of Second-Order ACGs

De Groote and Pogodalla [19] have shown that a variety of context-free formalisms can be encoded by second-order ACGs in a straightforward way.

[1] $\mathscr{L}_2(\mathsf{A})$ and $\mathscr{L}_2(\mathsf{E})$ are not represented as simply typed linear λ-terms here. Nevertheless, by introducing constants **Some** and **Every** of type $(e \to t)^2 \to t$, we can represent them by linear λ-terms as $\mathscr{L}_2(\mathsf{A}) = \lambda w_1 w_2.\mathbf{Some}\, w_1\, w_2$ and $\mathscr{L}_2(\mathsf{E}) = \lambda w_1 w_2.\mathbf{Every}\, w_1\, w_2$.

Namely, string languages generated by context-free grammars (CFGs), linear context-free grammars (LCFTGs), linear context-free rewriting systems (LCFRSs) are all generated by ACGs belonging to $\mathbf{G}_{\text{string}}(2,2)$, $\mathbf{G}_{\text{string}}(2,3)$, $\mathbf{G}_{\text{string}}(2,4)$ respectively. Here, we let the term *linear* CFTGs mean non-duplicating non-deleting CFTGs following the terminology by de Groote and Pogodalla, although usually "linearity" means just non-duplication. Their encodings are straightforward in the sense that the derivation structures in the original grammar are preserved in the resultant ACG. For the converse, it is not difficult to see that ACGs in $\mathbf{G}_{\text{string}}(2,2)$ can generate only context-free languages. The most remarkable result on the hierarchy of second-order string ACGs is given by Salvati [45]. The hierarchy collapses from fourth-order.

Theorem 2.6 (Salvati [45]). *Every second-order string ACG in* $\mathbf{G}_{\text{string}}(2,n)$ *for* $n \geq 1$ *has an equivalent LCFRS, and thus an equivalent ACG belonging to* $\mathbf{G}_{\text{string}}(2,4)$.

It is conjectured that the languages generated by ACGs in $\mathbf{G}_{\text{string}}(2,3)$ are also generated by LCFTGs. If that is the case, these results give us a precise correspondence between $\mathbf{G}_{\text{string}}(2,n)$ and context-free formalisms for each n.

Let us turn our attention to the generative capacity of second-order tree ACGs. The equivalence between $\mathbf{G}_{\text{tree}}(2,1)$ ($\mathbf{G}_{\text{tree}}(2,1(\text{r}))$) and regular tree grammars is easily seen. As discussed by de Groote and Pogodalla [19], linear context-free tree languages can be generated by ACGs in $\mathbf{G}_{\text{tree}}(2,2)$ (see also [27] for a complete proof). Moreover it is easy to see that the converse of the relation also holds. De Groote [18] has shown that tree-adjoining grammars are also simulated by ACGs in $\mathbf{G}_{\text{tree}}(2,2(\text{mon}))$. The converse relation also holds with respect to the string generating capacity.

Those correspondences are summarized in Table 2.1.

Definition 2.7 (Linear Context-Free Tree Grammar). A *context-free tree grammar* is a quadruple $G = \langle\langle V, \rho\rangle, T, R, S\rangle$ where $\langle V, \rho\rangle$ is a ranked alphabet, elements of $T \subsetneq V$ are called *terminal symbols*, elements of $V - T$ are called *nonterminal symbols*, $S \in V - T$ is the *start symbol* with $\rho(S) = 0$, and P is a finite set of *productions* of the form

$$A[x_1, \ldots, x_{\rho(A)}] \to N$$

where $A \in V - T$ and $N \in \mathbb{T}(\langle V, \rho\rangle \cup \{x_1, \ldots, x_{\rho(A)}\})$. A CFTG is *non-duplicating* iff for every production of the above form, x_i appears in N at most once for each $x_1, \ldots, x_{\rho(A)}$. A CFTG is *non-deleting* iff for every production

Table 2.1: Hierarchy of second-order ACGs

Finite Languages	$=$	$\mathbf{G}(1, m)$
String Languages		
Context-Free Grammars	$=$	$\mathbf{G}_{\text{string}}(2, 2)$
Linear Context Free Tree Grammars	$\subseteq$	$\mathbf{G}_{\text{string}}(2, 3)$
Linear Context Free Rewriting Systems	$=$	$\mathbf{G}_{\text{string}}(2, n)$ for $n \geq 4$
Tree Languages		
Regular Tree Grammars	$=$	$\mathbf{G}_{\text{tree}}(2, 1)$ ($\mathbf{G}_{\text{tree}}(2, 1(\text{r}))$)
Linear Context-Free Tree Grammars	$=$	$\mathbf{G}_{\text{tree}}(2, 2)$
Tree Adjoining Grammars	$\subseteq$	$\mathbf{G}_{\text{tree}}(2, 2(\text{mon}))$

rule of the above form, x_i appears in N at least once for each $x_1, \ldots, x_{\rho(A)}$. A CFTG is called *linear* iff it is non-duplicating and non-deleting at the same time.

We write

$$M \Rightarrow M'$$

iff there are $A \in V - T$, $M_0 \in \mathbb{T}(\langle V, \rho\rangle \cup \{z\})$ (z occurs exactly one in M_0), $M_1, \ldots, M_{\rho(A)} \in \mathbb{T}(\langle V, \rho\rangle)$, and $A[x_1, \ldots, x_{\rho(A)}] \to N \in R$ such that $M = M_0[AM_1 \ldots M_{\rho(A)}/z]$ and $M' = M_0[N[M_i/x_i]_{1 \leq i \leq \rho(A)}/z]$. The *(tree) language generated by* G is defined by

$$L(G) = \{\, M \in \mathbb{T}(\langle T, \rho\rangle) \mid S \stackrel{*}{\Rightarrow} M \,\}$$

where $\stackrel{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$.

De Groote and Pogodalla [19] encode an LCFTG $G = \langle\langle V, \rho\rangle, T, R, S\rangle$ as an ACG $\mathscr{G}^G = \langle \Sigma_0, \Sigma_{\langle T,\rho\rangle}, \mathscr{L}, S\rangle$ as follows. The object vocabulary is the tree signature $\Sigma_{\langle T,\rho\rangle}$, and the set $\mathscr{A}_0$ of abstract atomic types is the set $V - T$ of nonterminals. An abstract atomic type $A \in \mathscr{A}_0 = V - T$ is mapped to $o^{\rho(A)} \to o$ by the lexicon $\mathscr{L}$. For each production rule $r : A[x_1, \ldots, x_{\rho(A)}] \to N$, we put the following lexical entry into $\mathscr{G}^G$:

$$\langle \mathsf{A}_r,\ B_1 \to \cdots \to B_k \to A,\ \lambda y_1 \ldots y_k x_1 \ldots x_{\rho(A)}.N' \rangle$$

where $B_1, \ldots, B_k$ is a sequence of nonterminals occurring in N (if the same nonterminal B appears in N m-times, B occurs in the sequence m-times) and N' is obtained from N by replacing each B_i with y_i.

Example 2.8. Let an LCFTG G consist of the following production rules:

$$\begin{aligned} S &\to A\mathsf{abc}, \\ A[x_1, x_2, x_3] &\to A(\mathsf{fa}x_1)(\mathsf{fb}x_2)(\mathsf{fc}x_3), \\ A[x_1, x_2, x_3] &\to \mathsf{g}x_1x_2x_3, \end{aligned}$$

where the ranks of S, A, a, b, c, f, g, are 0, 3, 0, 0, 0, 2, 3, respectively. S and A are nonterminals and $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{f}, \mathsf{g}$ are terminals. The ACG $\mathscr{G}^G$ that encodes G consists of the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}(\mathsf{x})$
A_1	$A \to S$	$\lambda y_A^{o^3 \to o}.y_A\mathsf{abc}$
A_2	$A \to A$	$\lambda y_A^{o^3 \to o} x_1^o x_2^o x_3^o.y_A(\mathsf{fa}x_1)(\mathsf{fb}x_2)(\mathsf{fc}x_3)$
A_3	A	$\lambda x_1^o x_2^o x_3^o.\mathsf{g}x_1x_2x_3$

Definition 2.9 (Linear Context-Free Rewriting System). A *context-free rewriting system (CFRS)* is a quadruple $G = \langle\langle V, \rho\rangle, T, R, S\rangle$ where $\langle V, \rho\rangle$ is a ranked alphabet called the set of *nonterminals*, T is an (unranked) alphabet disjoint from V called the set of *terminal symbols*, $S \in V$ is the *start symbol* with $\rho(S) = 1$, and R is a finite set of *productions* of the form

$$A \to f(B_1, \ldots, B_m)$$

where $A, B_1, \ldots, B_m \in V$ and f is a function from $(T^*)^{\rho(B_1)} \times \cdots \times (T^*)^{\rho(B_m)}$ to $(T^*)^{\rho(A)}$ such that

$$f(\langle x_{1,1}, \ldots, x_{1,\rho(B_1)}\rangle, \ldots, \langle x_{m,1}, \ldots, x_{m,\rho(B_m)}\rangle) = \langle \mathsf{w}_1, \ldots, \mathsf{w}_{\rho(A)}\rangle$$

with $\mathsf{w}_i \in (T \cup \{x_{1,1}, \ldots, x_{m,\rho(B_m)}\})^*$. For f as above, f_i for $i \in \{1, \ldots, \rho(A)\}$ means the i-th project of f, i.e., $f_i(\vec{x}_1, \ldots, \vec{x}_m) = \mathsf{w}_i$ for $\vec{x}_i = \langle x_{i,1}, \ldots, x_{i,\rho(i)}\rangle$. A CFRS is *linear* iff for every function appearing in a production rule as the above form, $x_{i,j}$ appears in $f(\vec{x}_1, \ldots, \vec{x}_m)$ exactly once for each $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, \rho(B_i)\}$.

The set $L(G, A) \subseteq (T^*)^{\rho(A)}$ for $A \in V$ is defined as follows:

- If $A \to f() \in R$ (f is a constant function and $f() \in (T^*)^{\rho(A)}$), then $f() \in L(G, A)$.
- if $A \to f(B_1, \ldots, B_m) \in R$, and $\vec{\mathsf{w}}_i \in L(G, B_i)$ for $i \in \{1, \ldots, m\}$, then $f(\vec{\mathsf{w}}_1, \ldots, \vec{\mathsf{w}}_m) \in L(G, A)$.

The *language generated by* G is defined as

$$L(G) = \{\, \mathsf{w} \in T^* \mid \langle \mathsf{w}\rangle \in L(G, S) \,\}.$$

De Groote and Pogodalla [19] encode an LCFRS $G = \langle\langle V, \rho\rangle, T, R, S\rangle$ as an ACG $\mathscr{G}^G = \langle \Sigma_0, \Sigma_T, \mathscr{L}, s'\rangle$ as follows. The object vocabulary is the string signature Σ_T, and the set $\mathscr{A}_0$ of abstract atomic types is $V \cup \{s'\}$ with $s' \notin V$. An abstract atomic type $A \in V$ is mapped to $(\mathit{str}^{\rho(A)} \to \mathit{str}) \to \mathit{str}$ and the distinguished type s' is mapped to str by the lexicon $\mathscr{L}$. For each production rule $r : A \to f(B_1, \ldots, B_k)$, we put a lexical entry representing the rule:

$$\langle \mathsf{c}_r,\ B_1 \to \cdots \to B_k \to A,\ \lambda y_1 \ldots y_k z.y_1(\lambda \vec{x}_1. \ldots y_k(\lambda \vec{x}_k.zM_1 \ldots M_{\rho(A)}) \ldots)\rangle$$

where

$$M_i = /f_i(\vec{x}_1, \ldots, \vec{x}_k)/.$$

Moreover, we put the following special lexical entry:

$$\langle \mathsf{S},\ S \to s',\ \lambda y^{(\mathit{str} \to \mathit{str}) \to \mathit{str}}.y(\lambda \vec{x}^{\mathit{str}}.x)\rangle.$$

Then, whenever $\langle \mathsf{w}_1, \ldots, \mathsf{w}_{\rho(A)}\rangle \in L(G, A)$, we can find an abstract term $M \in \Lambda(\Sigma_0)$ of type A such that $\mathscr{L}(\mathsf{A}) = \lambda z^{(\mathit{str}^{\rho(A)} \to \mathit{str}) \to \mathit{str}}.z/\mathsf{w}_1/ \ldots /\mathsf{w}_{\rho(A)}/$.

Example 2.10. Let an LCFRS G consist of the following production rules:

$S \to f(A)$	with	$f(\langle x_1, x_2, x_3\rangle) = \langle x_1 x_2 x_3\rangle$
$A \to g(A)$	with	$g(\langle x_1, x_2, x_3\rangle) = \langle \mathsf{a}x_1, \mathsf{b}x_2, \mathsf{c}x_3\rangle$
$A \to h()$	with	$h() = \langle \mathsf{a}, \mathsf{b}, \mathsf{c}\rangle$

The ACG that encodes G consists of the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}(\mathsf{x})$
F	$A \to S$	$\lambda y_A z.y_A(\lambda x_1 x_2 x_3.z(x_1 + x_2 + x_3))$
G	$A \to A$	$\lambda y_A z.y_A(\lambda x_1 x_2 x_3.z(\mathsf{a} + x_1)(\mathsf{b} + x_2)(\mathsf{c} + x_3))$
H	A	$\lambda z.z\mathsf{abc}$
S	$S \to s'$	$\lambda y_S.y_S(\lambda x.x)$

A more general result on the generative capacity of second-order ACG is given by Salvati.

Theorem 2.11 (Salvati [43]). *Every second-order ACG generates a PTIME language.*

2.2.2 General Properties of ACGs

There are some results on the mathematical properties of ACGs which are themselves, or are corollaries to, already known facts. These properties demonstrate the rich expressive power of ACGs. However, some fundamental mathematical properties of ACGs are not yet to be revealed. For instance, no recursively enumerable language has been found that cannot be generated by any ACG.

Lemma 2.12. *Let a type substitution $\sigma : \{o\} \to \mathcal{T}(\{o\})$ be defined as $\sigma(o) = \mathit{str}$. Then, for any $\alpha \in \mathcal{T}(\{o\})$, there is a linear combinator $Z^{\sigma(\alpha)}$ of type $\sigma(\alpha)$.*

Proof. By induction on α, we define a linear combinator $Z^{\sigma(\alpha)}$ of type $\sigma(\alpha)$. Let $\sigma(\alpha) = \beta_1 \to \cdots \to \beta_m \to \mathit{str}$ and $\beta_i = \beta_{i,1} \to \cdots \to \beta_{i,k_i} \to \mathit{str}$. Define

$$Z^{\sigma(\alpha)} = \lambda y_1^{\beta_1} \ldots y_m^{\beta_m} z^o . R_1(R_2(\ldots(R_m z)\ldots))$$
$$\text{where } R_i = y_i^{\beta_i} Z^{\beta_{i,1}} \ldots Z^{\beta_{i,k_i}}.$$

□

Hereafter by Z^β we denote such linear combinator of type β provided that β has the above form.

Definition 2.13. The *universal membership problem* is the problem of determining whether $P \in \mathcal{O}(\mathscr{G})$. The *non-emptiness problem* for ACGs is the problem of determining whether $\mathcal{O}(\mathscr{G}) \neq \varnothing$.

Though these problems are fundamental, they are still open. We have only partial answers to them.

Proposition 2.14. *The universal membership problem for ACGs is at least as hard as the non-emptiness problem for ACGs.*

Proof. We present a reduction from the non-emptiness problem to the universal membership problem. Let an ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$ be given as an instance of the non-emptiness problem. We define $\mathscr{G}' = \langle \Sigma_0, \Sigma_1', \mathscr{L}', s \rangle$ as

$$\Sigma_1' = \langle \{o\}, \varnothing, \varnothing \rangle,$$
$$\mathscr{L}'(p) = \mathit{str} \text{ for all } p \in \mathscr{A}_0,$$
$$\mathscr{L}'(\mathsf{c}) = Z^{\mathscr{L}'(\tau_0(\mathsf{c}))} \text{ for all } \mathsf{c} \in \mathscr{C}_0,$$

where Z^α is as in Lemma 2.12. Then, $\mathscr{L}'(M) \twoheadrightarrow_\beta \lambda z^o.z$ for every $M \in \mathcal{A}(\mathscr{G}') = \mathcal{A}(\mathscr{G})$. Therefore, $\mathcal{O}(\mathscr{G}) \neq \varnothing$ iff $\lambda z^o.z \in \mathcal{O}(\mathscr{G}')$. □

The following proposition is an easy corollary to a result obtained by de Groote et al. [17]

Proposition 2.15. *The emptiness problem for ACGs is decidable iff the multiplicative exponential linear logic (***MELL***) is decidable.*

Proof. Let $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s\rangle$ be an ACG, where $\Sigma_0 = \langle \mathscr{A}_0, \mathscr{C}_0, \tau_0\rangle$, $\mathscr{C}_0 = \{\mathsf{c}_1, \ldots, \mathsf{c}_n\}$, and $\tau_0(\mathsf{c}_i) = A_i$ for $1 \leq i \leq n$. Then $\mathcal{O}(\mathscr{G}) \neq \varnothing$ iff $\mathcal{A}(\mathscr{G}) \neq \varnothing$ iff $!A_1, \ldots, !A_n \Rightarrow s$ is provable in **MELL**. This proves the "if" direction.

The "only if" direction can be proved as follows. De Groote et al. [17] show that the decidability of **MELL** is equivalent to the decidability of a fragment of it called $\mathbf{IMELL}_{\mathbf{0}}^{-\circ}$. Formulas of $\mathbf{IMELL}_{\mathbf{0}}^{-\circ}$ are of the form $!A$ or A, where A is a pure implicative formula, and the right-hand side of a sequent of $\mathbf{IMELL}_{\mathbf{0}}^{-\circ}$ must be a pure implicative formula. Given a sequent

$$\mathcal{S} : !A_1, \ldots, !A_n, B_1, \ldots, B_m \Rightarrow C$$

of $\mathbf{IMELL}_{\mathbf{0}}^{-\circ}$ where each B_i is pure implicative, let an ACG $\mathscr{G}^{\mathcal{S}} = \langle \Sigma_0, \Sigma_0, \mathscr{L}_{\mathrm{id}}, s\rangle$ be such that $\Sigma_0 = \langle \mathscr{A}_{\mathcal{S}} \cup \{s\}, \mathscr{C}_0, \tau_0\rangle$, $\mathscr{A}_{\mathcal{S}}$ is the set of atomic formulas in the sequent $\mathcal{S}$, $s \notin \mathscr{A}_{\mathcal{S}}$, $\mathscr{C}_0 = \{\mathsf{c}_i \mid 1 \leq i \leq n\} \cup \{\mathsf{b}\}$, $\tau_0(\mathsf{c}_i) = A_i$, $\tau_0(\mathsf{b}) = (B_1 \to \cdots \to B_m \to C) \to s$, and $\mathscr{L}_{\mathrm{id}}$ is the identity. Then $\mathcal{S}$ is provable in $\mathbf{IMELL}_{\mathbf{0}}^{-\circ}$ iff $\mathcal{S}$ is provable in **MELL** iff $\mathcal{A}(\mathscr{G}^{\mathcal{S}}) \neq \varnothing$ iff $\mathcal{O}(\mathscr{G}^{\mathcal{S}}) \neq \varnothing$. □

Restricting to string ACGs does not change the situation. We can find a string ACG $\mathscr{G}' = \langle \Sigma_0, \Sigma_1, \mathscr{L}', s\rangle$ such that $\mathcal{A}(\mathscr{G}') = \mathcal{A}(\mathscr{G}^{\mathcal{S}})$. For instance, let $\Sigma_1 = \langle \{o\}, \varnothing, \varnothing\rangle$, $\mathscr{L}'(p) = \mathit{str}$ for all $p \in \mathscr{A}_{\mathcal{S}} \cup \{s\}$ and let $\mathscr{L}'(\mathsf{c}) = Z^{\mathscr{L}(\tau_0(\mathsf{c}))}$ for all $\mathsf{c} \in \mathscr{C}_0$.

The decidability of **MELL** is still open but it is known to be at least as hard as Petri-net reachability, which is at least EXPSPACE-hard [31]. Indeed, we can represent Petri-net reachability sets by ACGs directly using a reduction from *vector addition systems (VASs)*, which are equivalent to Petri-nets.

Definition 2.16. An *m-dimensional vector addition system (m-VAS)* $\mathscr{V}$ is a pair $\langle \Delta, \vec{s}\rangle$, where Δ is a finite subset of $\mathbb{Z}^m$ and $\vec{s} \in \mathbb{N}^m$. We call $\mathbb{N}^m$ the set of *configurations*. $\Rightarrow_{\vec{d}}$ for $\vec{d} \in \mathbb{Z}^m$ is a binary relation on $\mathbb{N}^m$ such that for $\vec{a}, \vec{b} \in \mathbb{N}^m$, $\vec{a} \Rightarrow_{\vec{d}} \vec{b}$ iff $\vec{b} = \vec{a} + \vec{d}$. The union of $\Rightarrow_{\vec{d}}$ for $\vec{d} \in \Delta$ is denoted by $\Rightarrow_{\Delta}$. A configuration $\vec{b}$ is *reachable* iff $\vec{s} \Rightarrow_{\Delta}^{*} \vec{b}$ where $\Rightarrow_{\Delta}^{*}$ is the reflexive, transitive closure of $\Rightarrow_{\Delta}$. The *reachability set* $\mathcal{R}(\mathscr{V}) \subseteq \mathbb{N}^m$ *of an m-VAS* $\mathscr{V}$ is the set of reachable configurations.

Definition 2.17. For an alphabet V, suppose that its elements are ordered as $\mathsf{a}_1, \ldots, \mathsf{a}_n$. The *Parikh vector* of a string $\mathsf{w} \in V^*$ is defined as

$$\#(\mathsf{w}) = \langle \#_{\mathsf{a}_1}(\mathsf{w}), \ldots, \#_{\mathsf{a}_n}(\mathsf{w})\rangle \in \mathbb{N}^n.$$

and the *Parikh image* of a language $L \subseteq V^*$ is defined as

$$\#(L) = \{\#(\mathsf{w}) \mid \mathsf{w} \in L\} \subseteq \mathbb{N}^n.$$

A set of vectors of natural numbers $\Gamma \subseteq \mathbb{N}^m$ is *linear* iff there are $\vec{a}_1, \ldots, \vec{a}_n, \vec{b} \in \mathbb{N}^m$ such that $\Gamma = \{\vec{b} + \sum_{j=1}^{n} k_j \vec{a}_j \mid k_j \in \mathbb{N}\}$. $\Gamma \subseteq \mathbb{N}^m$ is *semilinear* iff there are linear sets $\Gamma_1, \ldots, \Gamma_n \subseteq \mathbb{N}^m$ such that $\Gamma = \bigcup_{i=1}^{n} \Gamma_i$. A language L is *linear (semilinear)* iff the Parikh image of L is linear (semilinear).

It is known that there is a VAS whose reachability set is not semilinear [21]. The notion of Parikh vector can be applied to λ-terms on a higher-order signature Σ; the *Parikh vector* of M is defined as $\#(M) = \langle \#_{\mathsf{a}_i}(M) \mid \mathsf{a}_i \in \mathscr{C} \rangle \in \mathbb{N}^{|\mathscr{C}|}$. Then the notions of Parikh image and the semilinearity of a set of terms are defined in the obvious way.

Proposition 2.18. *For every reachability set $\mathcal{R}(\mathscr{V})$, there is an ACG $\mathscr{G}^{\mathscr{V}} \in \mathbf{G}_{\mathrm{string}}(3,2)$ such that $\mathcal{R}(\mathscr{V}) = \#(\mathcal{O}(\mathscr{G}^{\mathscr{V}}))$.*

Proof. Given an m-VAS $\mathscr{V} = \langle \Delta, \vec{s} \rangle$, we define an ACG $\mathscr{G}^{\mathscr{V}} = \langle \Sigma_0, \Sigma_V, \mathscr{L}, q \rangle$ as follows. The set $\mathscr{A}_0$ of abstract atomic types is

$$\mathscr{A}_0 = \{\, p_1, \ldots, p_m, q \,\}.$$

Let a function ρ map $\vec{d} = \langle d_1, \ldots, d_m \rangle \in \mathbb{Z}^m$ to $\rho(\vec{d}) \in \mathcal{T}(\mathscr{A}_0)$ as

$$\rho(\vec{d}) = (p_1^{\nu(d_1)} \to \cdots \to p_m^{\nu(d_m)} \to q) \to p_1^{\pi(d_1)} \to \cdots \to p_m^{\pi(d_m)} \to q$$
$$\text{where } \nu(d_i) = \begin{cases} 0 & \text{if } d_i \geq 0 \\ -d_i & \text{otherwise} \end{cases}, \quad \pi(d_i) = \begin{cases} 0 & \text{if } d_i \leq 0 \\ d_i & \text{otherwise} \end{cases}$$

The set of abstract constants and the type assignment on them are defined as

$$\mathscr{C}_0 = \{\mathsf{a}_i \mid 1 \leq i \leq m\} \cup \{\mathsf{b}_{\vec{d}} \mid \vec{d} \in \Delta\} \cup \{\mathsf{c}\},$$
$$\tau_0(\mathsf{a}_i) = p_i, \quad \tau_0(\mathsf{b}_{\vec{d}}) = \rho(\vec{d}), \quad \tau_0(\mathsf{c}) = p_1^{s_1} \to \cdots \to p_m^{s_m} \to q,$$

where $\vec{s} = \langle s_1, \ldots, s_m \rangle$. The object vocabulary is the string signature Σ_V for the alphabet

$$V = \{\, \mathsf{e}_1, \ldots, \mathsf{e}_m \,\}.$$

The lexicon $\mathscr{L}$ is defined as

$$\mathscr{L}(p_i) = \mathscr{L}(q) = \mathit{str},$$
$$\mathscr{L}(\mathsf{a}_i) = \mathsf{e}_i, \quad \mathscr{L}(\mathsf{b}_{\vec{d}}) = Z^{\mathscr{L}(\rho(\vec{d}))}, \quad \mathscr{L}(\mathsf{c}) = Z^{\mathscr{L}(\tau_0(\mathsf{c}))}.$$

We show that $\vec{a} \in \mathcal{R}(\mathscr{V})$ iff there is $M \in \mathcal{A}(\mathscr{G}^{\mathscr{V}})$ such that $\#(\mathscr{L}(M)) = \vec{a}$.

First we show the "only if" part by induction on n where n is such that $\vec{s} \Rightarrow_{\Delta}^{n} \vec{a}$. For $n = 0$, i.e., $\vec{a} = \vec{s}$, let

$$M \equiv \mathsf{ca}_1^{s_1} \dots \mathsf{a}_m^{s_m} \in \mathcal{A}(\mathscr{G}^{\mathscr{V}}).$$

Suppose that

$$\vec{s} \Rightarrow_{\Delta}^{*} \vec{b} = \langle b_1, \dots, b_m \rangle \Rightarrow_{\vec{d}} \vec{a} = \langle a_1, \dots, a_m \rangle$$

for some $\vec{d} = \langle d_1, \dots, d_m \rangle \in \Delta$. By the induction hypothesis, we have $N \in \mathcal{A}(\mathscr{G}^{\mathscr{V}})$ such that $\#(\mathscr{L}(N)) = \vec{b}$. Since $a_i = b_i + d_i \geq 0$, we have $b_i \geq \nu(d_i)$ for every i. In other words, N contains at least $\nu(d_i)$ occurrences of the constant a_i for every i. Let N' be obtained from N by replacing $\nu(d_i)$ occurrences of the constant a_i with fresh distinct variables $x_{i,1}, \dots, x_{i,\nu(d_i)}$ for each i, i.e.,

$$N \equiv N'[\mathsf{a}_1/x_{1,1}, \dots, \mathsf{a}_1/x_{1,\nu(d_1)}, \dots, \mathsf{a}_m/x_{m,1}, \dots, \mathsf{a}_m/x_{m,\nu(d_m)}]$$

and N' contains $b_i - \nu(d_i)$ occurrences of a_i for each i. Let

$$M \equiv \mathsf{b}_{\vec{d}}(\lambda \vec{x}.N')\mathsf{a}_1^{\pi(d_1)} \dots \mathsf{a}_m^{\pi(d_m)} \in \mathcal{A}(\mathscr{G})$$

where $\vec{x} = \langle x_{1,1}, \dots, x_{1,\nu(d_1)}, \dots, x_{m,1}, \dots, x_{m,\nu(d_m)} \rangle$. Since M contains $a_i = b_i - \nu(d_i) + \pi(d_i) = b_i + d_i$ occurrences of a_i, we have $\#(\mathscr{L}(M)) = \vec{a}$.

Let $\mathscr{B} = \{\, \mathsf{b}_{\vec{d}} \in \mathscr{C}_0 \mid \vec{d} \in \Delta \,\}$. The "if" direction is shown by induction on $\#_{\mathscr{B}}(M)$ for $M \in \mathcal{A}(\mathscr{G}^{\mathscr{V}})$. M must be in the form of either

$$\mathsf{ca}_1^{s_1} \dots \mathsf{a}_m^{s_m}\ , \quad \text{or} \tag{2.1}$$

$$\mathsf{b}_{\vec{d}} N \mathsf{a}_1^{\pi(d_1)} \dots \mathsf{a}_m^{\pi(d_m)} \text{ for some } \vec{d} \in \Delta \text{ and } N \in \Lambda(\Sigma_0). \tag{2.2}$$

M has the form (2.1) iff $\#_{\mathscr{C}_{\mathsf{b}}}(M) = 0$. In this case, clearly $\#(\mathscr{L}(M)) = \vec{s}$ is reachable. If $\#_{\mathscr{B}}(M) \geq 1$, M has the form (2.2). Let $\vec{d} = \langle d_1, \dots, d_m \rangle$. By the type $\rho(\vec{d})$ of $\mathsf{b}_{\vec{d}}$,

$$\tau_0(N) = p_1^{\nu(d_1)} \to \dots \to p_m^{\nu(d_m)} \to q.$$

Let

$$N' \equiv N\mathsf{a}_1^{\nu(d_1)} \dots \mathsf{a}_m^{\nu(d_m)}$$

By applying the induction hypothesis to $N' \in \mathcal{A}(\mathscr{G}^{\mathscr{V}})$, we get that $\#(\mathscr{L}(N'))$ is reachable. Since $\#_{\mathsf{a}_i}(M) = \#_{\mathsf{a}_i}(N') - \nu(d_i) + \pi(d_i) = \#_{\mathsf{a}_i}(N') + d_i$, we have

$$\vec{s} \Rightarrow_{\Delta}^{*} \#(\mathscr{L}(N')) \Rightarrow_{\vec{d}} \#(\mathscr{L}(M)).$$

$\#(\mathscr{L}(M))$ is a reachable configuration. □

Corollary 2.19. *There is an ACG in* $\mathbf{G}(3,1)$ *whose language is not semilinear.*

Proof. We redefine $\mathscr{G}^{\curlyvee} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, q \rangle$ in the proof of Proposition 2.18 as $\mathscr{G}' = \langle \Sigma_0, \Sigma_0, \mathscr{L}_{\mathrm{id}}, q \rangle$, where $\mathscr{L}_{\mathrm{id}}$ is the identity. Since $\#(\mathcal{O}(\mathscr{G}^{\curlyvee}))$ can be obtained from $\#(\mathcal{A}(\mathscr{G}^{\curlyvee})) = \#(\mathcal{O}(\mathscr{G}'))$ by a projection, and semilinearity is preserved under projections, the non-semilinearity of the former implies the non-semilinearity of the latter. □

In contrast, the following result is easily seen.

Proposition 2.20. *Every second-order ACG generates a semilinear language.*

Proof. First we show that if an ACG $\mathscr{G}$ is second-order, then $\#(\mathcal{A}(\mathscr{G}))$ is semilinear. For the second-order abstract vocabulary Σ_0 of $\mathscr{G}$, let us define a CFG G^{Σ_0} so that the set of nonterminal symbol is $\mathscr{A}_0$, the set of terminal symbols is $\mathscr{C}_0$, the start symbol is the distinguished type s of $\mathscr{G}$, and the set of productions is

$$\{\, q \to \mathsf{A} p_1 \dots p_m \mid \mathsf{A} \in \mathscr{C}_0,\ \tau_0(\mathsf{A}) = p_1 \to \dots \to p_m \to q \,\}.$$

Clearly $\#(\mathcal{A}(\mathscr{G})) = \#(L(G^{\Sigma_0}))$, where $L(G^{\Sigma_0})$ denotes the language generated by G^{Σ_0}. Since every context-free language is semilinear, so is the abstract language of every second-order ACG.

Since the lexicon $\mathscr{L}$ induces a linear translation which maps $\#(M)$ to $\#(\mathscr{L}(M))$ for $M \in \Lambda(\Sigma_0)$ and semilinearity is preserved under linear translations, $\#(\mathcal{O}(\mathscr{G}))$ is also semilinear. □

Additionally, recent study by Kanazawa [26] shows that string ACLs form a *substitution-closed full abstract family of languages* in the sense of Ginsburg and Greibach [11]. Let $\mathbf{L}$ be a class of string languages. We say that $\mathbf{L}$ is a *full AFL* (*full abstract family of languages*) if $\mathbf{L}$ is closed under union, concatenation, Kleene star, homomorphism, inverse homomorphism, and intersection with regular sets. $\mathbf{L}$ is *closed under substitution* if, for every $L \in \mathbf{L}$ over an alphabet V_1 and every substitution f from V_1 to $\mathcal{P}(V_2^*)$ such that $f(\mathsf{a}) \in \mathbf{L}$ for all $\mathsf{a} \in V_1$, we have $\hat{f}(L) \in \mathbf{L}$, where $\mathcal{P}(V_2^*)$ denotes the power set of V_2^* and $\hat{f}$ is the extension of f such that $\hat{f}(\varepsilon) = \{\varepsilon\}$, $\hat{f}(\mathsf{wa}) = \hat{f}(\mathsf{w})\hat{f}(\mathsf{a})$ for $\mathsf{w} \in V_1^*$ and $\mathsf{a} \in V_1$, $\hat{f}(L) = \bigcup_{\mathsf{w} \in L} \hat{f}(\mathsf{w})$.

Theorem 2.21 (Kanazawa [26]). *The class of string languages generated by ACGs in* $\mathbf{G}_(m,n)$ *is a substitution-closed full AFL for all* $m, n \geq 2$.

Chapter 3

Non-Linear Extensions of Abstract Categorial Grammars

The theorems presented in Section 3.4 has been published in [60].

3.1 Introduction

The ACG formalism is based on simply typed linear lambda calculus in two senses,

(i) lexical entries of the grammar are all linear λ-terms,

(ii) grammatical combinations of them, i.e., elements of the abstract language, are also represented by linear λ-terms.

This accordance is, however, not mandatory. While the linearity constraint on the abstract language 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.

The ACG $\mathscr{G}_2$ in Example 2.5 can be improved by allowing non-linear terms. Let $\mathscr{G}_2'$ have the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}_2'(\mathsf{x})$
M	n	$\lambda x.\mathbf{man}\, x$
W	n	$\lambda x.\mathbf{woman}\, x$
J	np	$\mathbf{John}$
R	$np \to s$	$\lambda x.\mathbf{run}\, x$
L	$np^2 \to s$	$\lambda x_1 x_2.\mathbf{love}\, x_2\, x_1$
A	$n \to (np \to s) \to s$	$\lambda y_1 y_2.\exists(\lambda x.\wedge(y_1 x)(y_2 x))$
E	$n \to (np \to s) \to s$	$\lambda y_1 y_2.\forall(\lambda x.\to(y_1 x)(y_2 x))$

where $\mathscr{L}_2'(s) = t$, $\mathscr{L}_2'(np) = e$, $\mathscr{L}_2'(n) = e \to t$, and $\exists$, $\forall$, $\wedge$, $\longrightarrow$ are all object constants with $\tau_2'(\exists) = \tau_2'(\forall) = (e \to t) \to t$ and $\tau_2'(\wedge) = \tau_2'(\longrightarrow) = t^2 \to t$. The meanings of quantifiers A and E are represented by non-linear λ-terms which have multiple occurrences of bound variables,

Moreover, even vacuous λ-abstraction is useful for more flexible representations. Let the pair of two λ-terms M_1 and M_2 be represented by $\mathrm{Pair}M_1M_2 = \lambda v.vM_1M_2$ ($\mathrm{Pair}equiv\lambda u_1u_2v.vu_1u_2$). One can get the i-th projection of a given pair of λ-terms by the non-linear λ-term $\pi_i \equiv \lambda w.w(\lambda u_1u_2.u_i)$ as

$$\pi_i(\lambda v.vM_1M_2) \twoheadrightarrow_\beta (\lambda u_1u_2.u_i)M_1M_2 \to_\beta M_i.$$

For $\mathscr{G}_1$ consisting of the following lexical entries,

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}_1(\mathsf{x})$
M	n	$/\texttt{man}/$
W	n	$/\texttt{woman}/$
J	np	$/\texttt{John}/$
R	$np \to s$	$\lambda x.x + /\texttt{runs}/$
L	$np^2 \to s$	$\lambda x_1x_2.x_2 + /\texttt{loves}/ + x_1$
A	$n \to (np \to s) \to s$	$\lambda zw.w(/\texttt{a}/ + z)$
E	$n \to (np \to s) \to s$	$\lambda zw.w(/\texttt{every}/ + z)$

suppose that the word "`every`" is replaced with "`all`". Though the meanings of these two words are similar,

* `all man runs`

is not a correct sentence. The correct sentence is

`all men run`.

A modification $\mathscr{G}_1'$ is given as follows:

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}_1'(\mathsf{x})$
M	n	$P/\texttt{man}//\texttt{men}/$
W	n	$P/\texttt{woman}//\texttt{women}/$
J	np	$\lambda y.y/\texttt{John}/\pi_1$
R	$np \to s$	$\lambda x.x(\lambda uv.u + v(P/\texttt{runs}//\texttt{run}/))$
L	$np^2 \to s$	$\lambda x_1x_2.x_2(\lambda uv.u + v(P/\texttt{loves}//\texttt{love}/)) + x_1(\lambda uv.u)$
A	$n \to (np \to s) \to s$	$\lambda zw.w(\lambda y.y(/\texttt{a}/ + \pi_1z)\pi_1)$
E	$n \to (np \to s) \to s$	$\lambda zw.w(\lambda y.y(/\texttt{all}/ + \pi_2z)\pi_2)$

where $P = \lambda u_1^{\mathit{str}}u_2^{\mathit{str}}v^{\mathit{str}^2 \to \mathit{str}}.vu_1u_2$, $\pi_i \equiv \lambda w^{(\mathit{str}^2 \to \mathit{str}) \to \mathit{str}}.w(\lambda u_1^{\mathit{str}}u_2^{\mathit{str}}.u_i)$, $\mathscr{L}_1'(n) = (\mathit{str}^2 \to \mathit{str}) \to \mathit{str}$, $\mathscr{L}_1'(np) = (\mathit{str} \to ((\mathit{str}^2 \to \mathit{str}) \to \mathit{str}) \to \mathit{str}) \to \mathit{str}$,

$\mathscr{L}_1'(s) = \mathit{str}$. Then $\mathcal{O}(\mathscr{G}_1')$ consists of terms representing some English sentences such as `John runs`, `all men run`, `all women love a man`, and so on. The agreement between words is handled without using feature structures.

Both modifications on $\mathscr{G}_1$ and $\mathscr{G}_2$ are independently made without changing the abstract vocabulary Σ_0. The new ACGs $\mathscr{G}_1' = \langle \Sigma_0, \Sigma_1', \mathscr{L}_1', s \rangle$ and $\mathscr{G}_2' = \langle \Sigma_0, \Sigma_2', \mathscr{L}_2', s \rangle$ can generate correct pairs of a grammatical sentence and its meaning through the common abstract language.

On the other hand, it is known that the expressive power of some grammar formalisms involving a linearity constraint (non-duplication and non-deletion) does not change when the constraint is relaxed to just non-duplication, allowing deleting operations. Seki et al. [49] have shown the equivalence between linear context-free rewriting systems (LCFRSs) and multiple context-free grammars (MCFGs), which correspond to the relaxed version of LCFRSs that may have deleting operations. Fujiyoshi [10] has established the equivalence between linear (non-duplicating non-deleting) monadic CFTGs and non-duplicating monadic CFTGs. Fisher's result [8, 9] is rather general. He has shown that the string IO-languages defined by general CFTGs coincide with the string IO-languages defined by non-deleting CFTGs.

This chapter extends the definition of ACGs and moreover introduces variants of the definition of the abstract languages. A brief discussion on the relation among the resultant various classes of ACLs defined by those extensions is given in Section 3.3. In Section 3.4 we then focus on the relation between usual linear ACGs and *affine* ACGs, whose lexical entries may contain vacuous λ-abstraction, along the line mentioned above. We present a procedure for constructing a linear ACG corresponding to a given affine ACG such that the language of the constructed ACG is exactly the set of the linear λ-terms generated by the original ACG. Therefore, we conclude that affine ACGs are not essentially more expressive than linear ACGs, since strings and trees are usually represented by linear λ-terms.

As de Groote and Pogodalla [18,19] have constructed linear ACGs encoding linear CFTGs and LCFRSs, non-duplicating CFTGs and MCFGs are also encodable by affine ACGs 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.

3.2 Definitions

In this chapter, we deal with non-linear λ-terms, so the word "λ-term" does not mean "*linear* λ-term", but a general unrestricted λ-term belonging to $\Lambda^{\mathrm{K}}(\Sigma)$ defined in Chapter 2. To emphasize the absence of the restriction to linear λ-terms, we call them λK-terms. A λK-term is a λI-*term*, if for every subterm of M of the form $\lambda x.N$, $x \in \mathrm{Fv}(N)$. An *affine term* is a λK-term such that no variable occurs free twice or more in any subterm of M. Thus, a linear term is a λK-term that is λI and affine at the same time. We write $\Lambda^{\mathrm{K}}(\Sigma)$, $\Lambda^{\mathrm{I}}(\Sigma)$, $\Lambda^{\mathrm{aff}}(\Sigma)$, $\Lambda^{\mathrm{lin}}(\Sigma)$, respectively for the sets of simply typed λK-terms, λI-terms, affine terms, linear terms on Σ.

We extend the definition of a lexicon $\mathscr{L}$ from Σ_0 to Σ_1 so that $\mathscr{L}(\mathsf{a})$ is not restricted to linear terms but any simply typed λK-term on Σ_1 provided that it is a compatible pair of a type substitution and a term substitution. We call the generalized lexicons λK-*lexicons*. Similarly we define a λI-*lexicon*, an *affine lexicon* and a *linear lexicon*. A λK-*ACG*, a λI-*ACG*, an *affine ACG*, a *linear ACG* are defined in the same way as usual ACGs but the lexicons $\mathscr{L}$ are λK, λI, affine, linear, respectively. $\mathbf{G}^{\mathrm{K}}$, $\mathbf{G}^{\mathrm{I}}$, $\mathbf{G}^{\mathrm{aff}}$, $\mathbf{G}^{\mathrm{lin}}$ denotes the class of λK, λI, affine, linear ACGs respectively, so ACGs satisfying the original definition are called with the modifier "linear" in this chapter. For each $\mathbf{G}^* \in \{\mathbf{G}^{\mathrm{K}}, \mathbf{G}^{\mathrm{I}}, \mathbf{G}^{\mathrm{aff}}, \mathbf{G}^{\mathrm{lin}}\}$, the subclasses denoted by $\mathbf{G}^*(m,n) \subseteq \mathbf{G}^*$ are defined in the same way as in Chapter 2.

Moreover, as we have done for the definition of an ACG, we can give variations of the definitions of the *abstract language* and *object language* of an ACG. For a λK-ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$, let

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

Note that while $\mathcal{A}^{\mathrm{lin}}(\mathscr{G})$ is a subset of $\Lambda^{\mathrm{lin}}(\Sigma_0)$, $\mathcal{O}^{\mathrm{lin}}(\mathscr{G})$ is not necessarily a subset of $\Lambda^{\mathrm{lin}}(\Sigma_1)$ if $\mathscr{G} \notin \mathbf{G}^{\mathrm{lin}}$. We define $\mathcal{A}^{\mathrm{aff}}(\mathscr{G})$, $\mathcal{O}^{\mathrm{aff}}(\mathscr{G})$, $\mathcal{A}^{\mathrm{I}}(\mathscr{G})$, $\mathcal{O}^{\mathrm{I}}(\mathscr{G})$ similarly. For a class $\mathbf{G}^*$ of ACGs, let

$$\begin{aligned}
\mathbf{L}^{\mathrm{K}}(\mathbf{G}^*) &= \{\, \mathcal{O}^{\mathrm{K}}(\mathscr{G}) \mid \mathscr{G} \in \mathbf{G}^* \,\} \\
\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^*) &= \{\, \mathcal{O}^{\mathrm{lin}}(\mathscr{G}) \mid \mathscr{G} \in \mathbf{G}^* \,\}
\end{aligned}$$

and $\mathbf{L}^{\mathrm{aff}}(\mathbf{G}^*)$, $\mathbf{L}^{\mathrm{I}}(\mathbf{G}^*)$ be defined similarly. $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{lin}})$ is the class of original abstract categorial languages.

3.3 Relations among Variants of ACLs

By extending the definition of ACGs and their languages, we have obtained sixteen classes of languages. In this section, we discuss the relation among those classes. Let us focus on the four classes of abstract categorial languages $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{lin}})$, $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{aff}})$, $\mathbf{L}^{\mathrm{aff}}(\mathbf{G}^{\mathrm{lin}})$, and $\mathbf{L}^{\mathrm{aff}}(\mathbf{G}^{\mathrm{aff}})$. We can compare other classes of languages defined in the previous section similarly. First, by $\mathbf{G}^{\mathrm{lin}} \subseteq \mathbf{G}^{\mathrm{aff}}$, trivially $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{lin}}) \subseteq \mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{aff}})$ and $\mathbf{L}^{\mathrm{aff}}(\mathbf{G}^{\mathrm{lin}}) \subseteq \mathbf{L}^{\mathrm{aff}}(\mathbf{G}^{\mathrm{aff}})$ hold. Note that the fact that $\mathcal{O}^{\mathrm{lin}}(\mathscr{G}) \subseteq \mathcal{O}^{\mathrm{aff}}(\mathscr{G})$ for all $\mathscr{G} \in \mathbf{G}$ does not imply that $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{*}) \subseteq \mathbf{L}^{\mathrm{aff}}(\mathbf{G}^{*})$ for $\mathbf{G}^{*} \in \{\mathbf{G}^{\mathrm{lin}}, \mathbf{G}^{\mathrm{aff}}\}$. In fact, the following proposition is rather easy.

Proposition 3.1. $\mathbf{L}^{\mathrm{aff}}(\mathbf{G}^{\mathrm{aff}}(m,n)) \subseteq \mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{aff}}(m,n))$.

Proof. Given $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle \in \mathbf{G}^{\mathrm{aff}}$, let us define $\mathscr{G}' = \langle \Sigma_0', \Sigma_1, \mathscr{L}', s \rangle \in \mathbf{G}^{\mathrm{aff}}$ as follows:

$$
\begin{aligned}
\mathscr{A}_0' &= \mathscr{A}_0, \\
\mathscr{C}_0' &= \mathscr{C}_0 \cup \{\, \mathsf{K}_\alpha^\beta \mid \beta \text{ is a negative subtype of } \tau_0(\mathsf{b}) \text{ for some } \mathsf{b} \in \mathscr{C}_0, \text{ and} \\
&\qquad \alpha \text{ is a negative subtype of a negative subtype of } \tau_0(\mathsf{a}) \text{ for some } \mathsf{a} \in \mathscr{C}_0 \,\}, \\
\tau_0' &= \tau_0 \cup \{\, \mathsf{K}_\alpha^\beta \mapsto \beta \to \alpha \to \beta \,\}, \\
\mathscr{L}' &= \mathscr{L} \cup \{\, \mathsf{K}_\alpha^\beta \mapsto \lambda x^{\mathscr{L}(\beta)} y^{\mathscr{L}(\alpha)}.x \,\}.
\end{aligned}
$$

It is enough to show that $\mathcal{O}^{\mathrm{aff}}(\mathscr{G}) = \mathcal{O}^{\mathrm{lin}}(\mathscr{G}')$.

We first show that $\mathcal{O}^{\mathrm{aff}}(\mathscr{G}) \subseteq \mathcal{O}^{\mathrm{lin}}(\mathscr{G}')$. For $M \in \Lambda^{\mathrm{aff}}(\Sigma_0)$, let $(M)^*$ be defined as follows:

$$
\begin{aligned}
(M)^* &= M \quad \text{if } M \in \mathscr{C}_0 \cup \mathscr{X}, \\
(M_1 M_2)^* &= (M_1)^* (M_2)^*, \\
(\lambda x^\alpha.M)^* &= \begin{cases} \lambda x^\alpha.(M)^* & \text{if } x \in \mathrm{Fv}(M), \\ \mathsf{K}_\alpha^\beta (M)^* & \text{if } x \notin \mathrm{Fv}(M) \text{ and } \tau_0(M) = \beta. \end{cases}
\end{aligned}
$$

It is easy to see that for every $M \in \mathcal{A}^{\mathrm{aff}}(\mathscr{G})$, $(M)^* \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G}')$ and $\mathscr{L}'((M)^*) = \mathscr{L}(M)$.

Conversely, for $N \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G}')$, let M be the term obtained from N replacing each occurrence of a constant K_α^β with $\lambda x^\beta y^\alpha.x$. Clearly $M \in \mathcal{A}^{\mathrm{aff}}(\mathscr{G})$ and $\mathscr{L}'(N) = \mathscr{L}(M)$. □

Thus, $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{aff}})$ is the largest class among the four classes. The above proof transfers deleting operations in the abstract level into the object level via constants K that are mapped to $\lambda xy.x$. Therefore, for $\mathbf{G}^{*} \in \{\mathbf{G}^{\mathrm{aff}}, \mathbf{G}^{\mathrm{K}}\}$,

$$\begin{array}{ccc} \mathbf{L}^{\circ}(\mathbf{G}^{\circ}(m,n)) & \subseteq & \mathbf{L}^{\bullet}(\mathbf{G}^{\circ}(m,n)) \\ \cup\!| & & \cup\!| \\ \mathbf{L}^{\circ}(\mathbf{G}^{\bullet}(m,n)) & & \mathbf{L}^{\bullet}(\mathbf{G}^{\bullet}(m,n)) \end{array} \qquad \text{if } \bullet \prec \circ \text{ for } \begin{cases} \text{lin} \prec \text{aff} \prec \text{K}, \\ \text{lin} \prec \text{I} \prec \text{K}. \end{cases}$$

Figure 3.1: Relation among variants of ACLs

we get $\mathbf{L}^{\mathrm{K}}(\mathbf{G}^{*}(m,n)) \subseteq \mathbf{L}^{\mathrm{I}}(\mathbf{G}^{*}(m,n))$ and $\mathbf{L}^{\mathrm{aff}}(\mathbf{G}^{*}(m,n)) \subseteq \mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{*}(m,n))$. Similarly, duplicating operations in the abstract level can be relegated to the object level via constants W that are mapped to $\lambda xy.xyy$. For $\mathbf{G}^{*} \in \{\mathbf{G}^{\mathrm{I}}, \mathbf{G}^{\mathrm{K}}\}$, we get $\mathbf{L}^{\mathrm{K}}(\mathbf{G}^{*}(m,n)) \subseteq \mathbf{L}^{\mathrm{aff}}(\mathbf{G}^{*}(m,n))$ and $\mathbf{L}^{\mathrm{I}}(\mathbf{G}^{*}(m,n)) \subseteq \mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{*}(m,n))$. Figure 3.1 summarizes the relations among variants of ACLs.

Corollary 3.2. *Let* $*$ *be among* $\{\mathrm{K}, \mathrm{I}, \mathrm{aff}\}$. *The class* $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{*}(m,n))$ *is the largest among* $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{*}(m,n))$, $\mathbf{L}^{*}(\mathbf{G}^{*}(m,n))$, $\mathbf{L}^{*}(\mathbf{G}^{\mathrm{lin}}(m,n))$, $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{lin}}(m,n))$.

Therefore, relaxing the linearity constraint on the abstract language does not enlarge the class of ACLs as much as extending the definition of ACGs themselves. In the sequel, hence we focus only on ACLs constructed on linear abstract languages.

3.4 Linearization of Affine ACGs

In the remainder of this chapter, we investigate the relation between $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{lin}})$ and $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{aff}})$ in more detail. While $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{lin}})$ consists of languages whose elements are all linear, $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{aff}})$ contain languages including non-linear terms. Therefore, $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{aff}})$ is properly larger than $\mathbf{L}^{\mathrm{lin}}(\mathbf{G}^{\mathrm{lin}})$. However, since λ-terms representing strings or trees are all linear, non-linear terms in the object languages are not very interesting. The main result of this chapter is that for every $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}(m,n)$, we can construct $\mathscr{G}^{l} \in \mathbf{G}^{\mathrm{lin}}(m, \max\{2,n\})$ such that

$$\mathcal{O}^{\mathrm{lin}}(\mathscr{G}^{l}) = \{\, P \in \mathcal{O}^{\mathrm{lin}}(\mathscr{G}) \mid P \text{ is linear} \,\}. \tag{3.1}$$

Moreover, in case of $m = 2$, we can find $\mathscr{G}^{l} \in \mathbf{G}^{\mathrm{lin}}(2,n)$ satisfying the equation (3.1). Therefore extending the definition of an ACG to allow lexical entries to be 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 second-order string ACGs, obtained from Salvati's work [45]. He presents an algorithm that converts a linear second-order string ACG $\mathscr{G} \in \mathbf{G}^{\mathrm{lin}}_{\mathrm{string}}(2,n)$ into an equivalent LCFRS (via a deterministic tree-walking transducer). Even if an input is an affine ACG $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}_{\mathrm{string}}(2,n)$, his algorithm still outputs an equivalent LCFRS. Since every LCFRS is encodable by a linear ACG belonging to $\mathbf{G}^{\mathrm{lin}}_{\mathrm{string}}(2,4)$ [18,19], this entails the following corollary.

Corollary 3.3. *For every affine ACG $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}_{\mathrm{string}}(2,n)$, there is a linear ACG $\mathscr{G}' \in \mathbf{G}^{\mathrm{lin}}_{\mathrm{string}}(2,4)$ such that $\mathcal{O}^{\mathrm{lin}}(\mathscr{G}') = \mathcal{O}^{\mathrm{lin}}(\mathscr{G})$.*

3.4.1 Basic Idea

We explain our basic idea for 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 $\mathsf{AB} \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G})$, we have $\mathsf{a} \in \mathcal{O}^{\mathrm{lin}}(\mathscr{G})$ by

$$\mathscr{L}(\mathsf{AB}) \equiv (\lambda w^{o\to o\to o}.w\mathsf{a}^o\mathsf{b}^o)(\lambda x^o y^o.x) \to_\beta (\lambda x^o y^o.x)\mathsf{a}^o\mathsf{b}^o \twoheadrightarrow_\beta \mathsf{a}^o. \tag{3.2}$$

The occurrences of vacuous λ-abstraction λy^o causes the deletion of b in (3.2). 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 (3.2) is decorated by bars as

$$(\lambda w^{o\to\overline{o}\to o}.w\mathsf{a}^o\overline{\mathsf{b}}^{\overline{o}})(\lambda x^o y^{\overline{o}}.x) \to_\beta (\lambda x^o y^{\overline{o}}.x)\mathsf{a}^o\overline{\mathsf{b}}^{\overline{o}} \twoheadrightarrow_\beta \mathsf{a}^o, \tag{3.3}$$

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 (3.3), we get

$$(\lambda w^{o\to o}.w\mathsf{a}^o)(\lambda x^o.x) \to_\beta (\lambda x^o.x)\mathsf{a}^o \to_\beta \mathsf{a}^o, \tag{3.4}$$

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 $\mathscr{L}(\mathsf{AB}) = \mathscr{L}'(\mathsf{A'B'})$. 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 subterm $\overline{\mathsf{a}}$ 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.

3.4.2 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

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

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 relabeling lexicon $\widetilde{\cdot}$ from Σ_1' to Σ_1 defined as

$$\begin{cases} \widetilde{\bar{p}} = \widetilde{p} = p & \text{for } p \in \mathscr{A}_1, \\ \widetilde{\bar{\mathsf{c}}} \equiv \widetilde{\mathsf{c}} \equiv \mathsf{c} & \text{for } \mathsf{c} \in \mathscr{C}_1. \end{cases}$$

For $P \in \Lambda^{\mathrm{aff}}(\Sigma_1)$, we let $\overline{P}$ denote the unique term such that $\overline{P} \in \Lambda^{\mathrm{aff}}(\overline{\Sigma_1})$ and $\widetilde{\overline{P}} \equiv P$.

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 subterm of it. For instance, if a variable x has type $o \to \bar{o} \notin \widehat{\mathcal{T}}(\{o\})$, then the term $x^{o \to \bar{o}}y^o$ has type $\bar{o}$, which, in our setting, means that it should disappear. But if $x^{o \to \bar{o}}y^o$ disappears, so does y^o, which, therefore, should have type $\bar{o}$ 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 subterm of Q and $x^\alpha \notin \mathrm{Fv}(Q')$, 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 $Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$ has a type in $\mathcal{T}(\overline{\mathscr{A}_1})$, then every subterm 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 $P \in \Lambda^{\mathrm{aff}}(\Sigma_1)$, Π gives the set of possible bar-decorations on them:

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

In other words, Π 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 subterms 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:

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

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 3.4. *Let $Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$ be given. If Q and its free variables have types in $\mathcal{T}(\mathscr{A}_1)$, then $(Q)^\dagger =_\beta Q =_\beta \widetilde{Q}$.*

Proof. Since both functions $\widetilde{\cdot}$ and $\dagger$ preserve the β-equality, we can assume that Q is β-normal. We show that $(Q)^\dagger \equiv Q \equiv \widetilde{Q}$ by induction on Q.

If $Q \equiv x^\alpha Q_1 \dots Q_m$, by $\alpha \in \mathcal{T}(\mathscr{A}_1)$, we have $\tau_1'(Q_i) \in \mathcal{T}(\mathscr{A}_1)$ for each $i \in \{1, \dots, m\}$. Applying the induction hypothesis to each Q_i, we get the conclusion $(Q)^\dagger \equiv Q \equiv \widetilde{Q}$.

If $Q \equiv \mathsf{c} Q_1 \dots Q_m$ for $\mathsf{c} \in \mathscr{C}_1'$, by $\tau_1'(Q) \in \mathcal{T}(\mathscr{A}_1)$, we have $\mathsf{c} \in \mathscr{C}_1$. By the type of c, we can apply the induction hypothesis to each Q_i, and get the conclusion $(Q)^\dagger \equiv Q \equiv \widetilde{Q}$.

If $Q \equiv \lambda x^\alpha.Q'$, by $\tau_1'(Q) = \alpha \to \tau_1'(Q') \in \mathcal{T}(\mathscr{A}_1)$, we can apply the induction hypothesis to Q'. Thus $(Q)^\dagger \equiv Q \equiv \widetilde{Q}$. □

Lemma 3.5. *For every closed term $P \in \Lambda^{\mathrm{aff}}(\Sigma_1)$, $|P|_\beta$ is linear iff there is $Q \in \Pi(P)$ whose type is in $\mathcal{T}(\mathscr{A}_1)$.*

Proof. First we show the "if" direction. If there is $Q \in \Pi(P)$ with $\tau_1'(Q) \in \mathcal{T}(\mathscr{A}_1)$, then $(Q)^\dagger =_\beta \widetilde{Q} \equiv P$ by Lemma 3.4. Since $(Q)^\dagger$ is linear, so is $|P|_\beta$.

Second we show the "only if" direction by induction on the maximum number of β-reduction steps from P to $|P|_\beta$. If P is β-normal and linear, then $P \in \Pi(P)$. Otherwise, suppose that P has a β-redex as $P \equiv P_0[z := (\lambda x^\alpha.P_1)P_2]$, where z occurs free in P_0.

Case 1. $x^\alpha \in \mathrm{Fv}(P_1)$ and

$$P \equiv P_0[z := (\lambda x^\alpha.P_1)P_2] \to_\beta P_0[z := P_1[P_2/x^\alpha]] \twoheadrightarrow_\beta |P|_\beta.$$

By the induction hypothesis, there is $Q' \in \Pi(P_0[z := P_1[P_2/x^\alpha]])$ such that $\tau_1'(Q') \in \mathcal{T}(\mathscr{A}_1)$. Q' has the form $Q' \equiv Q_0[z := Q_1[Q_2/x^\beta]]$ for $Q_i \in \Pi(P_i)$ for $i = 0, 1, 2$, and $\beta \in \Pi(\alpha)$. Let $Q \equiv Q_0[z := (\lambda x^\beta.Q_1)Q_2]$. Clearly $Q \in \Pi(P)$ and $\tau_1'(Q) = \tau_1'(Q') \in \mathcal{T}(\mathscr{A}_1)$.

Case 2. $x^\alpha \notin \mathrm{Fv}(P_1)$ and

$$P \equiv P_0[z := (\lambda x^\alpha.P_1)P_2] \to_\beta P_0[z := P_1] \twoheadrightarrow_\beta |P|_\beta.$$

By the induction hypothesis, there is $Q' \in \Pi(P_0[z := P_1])$ such that $\tau_1'(Q') \in \mathcal{T}(\mathscr{A}_1)$. Q' has the form $Q' \equiv Q_0[z := Q_1]$ for $Q_i \in \Pi(P_i)$ for $i = 0, 1$. Suppose that $y^\gamma \in \mathrm{Fv}(P_2)$. Since $P_0[z := (\lambda x^\alpha.P_1)P_2]$ is closed, P_0 contains an occurrence of the λ-abstraction λy^γ that binds the free occurrence of y^γ

in P_2. The occurrence of the λ-abstraction λy^{γ} becomes vacuous in $P_0[z := P_1]$. By $Q' \in \Pi(P_0[z := P_1]) \subseteq \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$, the corresponding occurrence of the λ-abstraction in Q_0 is $\lambda y^{\bar{\gamma}}$. Let $Q \equiv Q_0[z := (\lambda x^{\bar{\alpha}}.Q_1)\overline{P_2}]$. By the above observation, Q is well-typed (the types of bound variables and their abstractions are consistent). Clearly $Q \in \Pi(P)$ and $\tau_1'(Q) = \tau_1'(Q') \in \mathcal{T}(\mathscr{A}_1)$. □

Second-Order Case

We say that an abstract atomic type $p \in \mathscr{A}_0$ is *useless* if there is no $M \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G})$ that has a subterm whose type contains p. An abstract constant a is *useless* if there is no $M \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G})$ 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 easily eliminate useless abstract atomic types and constants. This can be done as elimination of useless nonterminals and productions from a context-free grammar.

Definition 3.6. 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 useless abstract constants. We define $\mathscr{G}' = \langle \Sigma_0', \Sigma_1, \mathscr{L}', [s, \mathscr{L}(s)] \rangle$ as follows: define Σ_0' by

$$\begin{aligned}
\mathscr{A}_0' &= \{\, [p, \beta] \mid p \in \mathscr{A}_0,\ \beta \in \Pi(\mathscr{L}(p)) - \mathcal{T}(\overline{\mathscr{A}_1}) \,\} \\
\mathscr{C}_0' &= \{\, [\![\mathsf{a}, Q]\!] \mid \mathsf{a} \in \mathscr{C}_0,\ Q \in \Pi(\mathscr{L}(\mathsf{a})) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_1}) \,\} \\
\tau_0' &= \{\, [\![\mathsf{a}, Q]\!] \mapsto ([\tau_0(\mathsf{a}), \tau_1'(Q)])^{\ddagger} \,\} \text{ where} \\
&\quad ([p, \beta])^{\ddagger} = [p, \beta], \\
&\quad ([\alpha \to \gamma, \beta \to \delta])^{\ddagger} = \begin{cases} ([\alpha, \beta])^{\ddagger} \to ([\gamma, \delta])^{\ddagger} & \text{if } \beta \notin \mathcal{T}(\overline{\mathscr{A}_1}), \\ ([\gamma, \delta])^{\ddagger} & \text{if } \beta \in \mathcal{T}(\overline{\mathscr{A}_1}), \end{cases}
\end{aligned}$$

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 3.7. *Let $\mathscr{G}$ and $\mathscr{G}'$ be as in Definition 3.6.*

For every variable-free $M \in \Lambda^{\mathrm{lin}}(\Sigma_0)$ of an atomic type and every $Q \in \Pi(\mathscr{L}(M)) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_1})$, 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 $Q \in \Pi(\mathscr{L}(M)) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_1})$ such that $\tau_0'(N) = [\tau_0(M), \tau_1'(Q)]$ and $\mathscr{L}'(N) \equiv (Q)^{\dagger}$.

Proof. Induction on M and N respectively.

Let $M = \mathsf{a}M_1 \dots M_m \in \Lambda^{\mathrm{lin}}(\Sigma_0)$. $Q \in \Pi(\mathscr{L}(M))$ is written as

$$Q \equiv Q_0 Q_1 \dots Q_m,$$

where $Q_0 \in \Pi(\mathscr{L}(\mathsf{a}))$ and $Q_i \in \Pi(\mathscr{L}(M_i))$ for $i \in \{1, \dots, m\}$. Let $I = \{\, i \mid 1 \leq i \leq m,\ \tau_1'(Q_i) \notin \mathcal{T}(\overline{\mathscr{A}_1}) \,\}$. Then,

$$(Q)^\dagger \equiv (Q_0)^\dagger (Q_{i_1})^\dagger \dots (Q_{i_k})^\dagger,$$

where $\{i_1, \dots, i_k\} = I$. By the definition of $\mathscr{G}'$, there is $[\![\mathsf{a}, Q_0]\!] \in \mathscr{C}_0'$ such that

$$\tau_0'([\![\mathsf{a}, Q_0]\!]) = [\tau_0(M_{i_1}), \tau_1'(Q_{i_1})] \to \dots \to [\tau_0(M_{i_k}), \tau_1'(Q_{i_k})] \to [\tau_0(M), \tau_1'(Q)]$$
$$\mathscr{L}'([\![\mathsf{a}, Q_0]\!]) \equiv (Q_0)^\dagger$$

For each $i \in I$, the induction hypothesis gives $N_i \in \Lambda^{\mathrm{lin}}(\Sigma_0')$ such that $\tau_0'(N_i) = [\tau_0(M_i), \tau_1'(Q_i)]$ and $\mathscr{L}'(N_i) \equiv (Q_i)^\dagger$. Clearly for

$$N \equiv [\![\mathsf{a}, Q_0]\!] N_{i_1} \dots N_{i_k},$$

we have $\tau_0'(N) = [\tau_0(M), \tau_1'(Q)]$ and $\mathscr{L}'(N) \equiv (Q)^\dagger$.

Conversely, suppose that a variable free term $N \in \Lambda^{\mathrm{lin}}(\Sigma_0')$ of an atomic type is given. For the head $[\![\mathsf{a}, Q_0]\!]$ of N, let

$$\tau_0(\mathsf{a}) = p_1 \to \dots \to p_m \to p_0,$$
$$\tau_1'(Q_0) = \alpha_1 \to \dots \to \alpha_m \to \alpha_0,$$

where $\alpha_i \in \Pi(\mathscr{L}(p_i))$ for $i \in \{0, 1, \dots, m\}$. Let $I = \{\, i \mid 1 \leq i \leq m,\ \alpha_i \notin \mathcal{T}(\overline{\mathscr{A}_1}) \,\}$. Then, we have

$$\tau_0'([\![\mathsf{a}, Q_0]\!]) = [p_{i_1}, \alpha_{i_1}] \to \dots \to [p_{i_k}, \alpha_{i_k}] \to [p_0, \alpha_0],$$
$$N \equiv [\![\mathsf{a}, Q_0]\!] N_{i_1} \dots N_{i_k},$$

where $\{i_1, \dots, i_k\} = I$. For each $i \in I$, the induction hypothesis gives $M_i \in \Lambda^{\mathrm{lin}}(\Sigma_0)$ and $Q_i \in \Pi(\mathscr{L}(M_i)) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_1})$ such that $\tau_0'(N_i) = [\tau_0(M_i), \tau_1'(Q_i)] = [p_i, \alpha_i]$ and $\mathscr{L}'(N_i) \equiv (Q_i)^\dagger$. Recall that $\mathscr{G}$ has neither useless atomic types nor useless constants. Thus for each $j \in \{1, \dots, m\} - I$, we can find a variable-free term $M_j \in \Lambda^{\mathrm{lin}}(\Sigma_0)$ of type p_j. Let $Q_j \equiv \overline{\mathscr{L}(M_j)}$ for $j \in \{1, \dots, m\} - I$. Let

$$M \equiv \mathsf{a}M_1 \dots M_m,$$
$$Q \equiv Q_0 Q_1 \dots Q_m.$$

Clearly

$$\begin{aligned} Q &\in \Pi(\mathscr{L}(M)) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_0}), \\ \tau_0'(N) &= [p_0, \alpha_0] = [\tau_0(M), \tau_1'(Q)], \\ \mathscr{L}'(N) &\equiv \mathscr{L}'([\![\mathsf{a}, Q_0]\!] N_{i_1} \dots N_{i_k}) \\ &\equiv (Q_0)^\dagger (Q_{i_1})^\dagger \dots (Q_{i_k})^\dagger \\ &\equiv (Q)^\dagger. \end{aligned}$$

□

Theorem 3.8. *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 $\mathcal{O}^{\mathrm{lin}}(\mathscr{G}^l) = \{\, P \in \mathcal{O}^{\mathrm{lin}}(\mathscr{G}) \mid P \text{ is linear} \,\}$.*

Proof. It is enough to show that $\mathcal{O}^{\mathrm{lin}}(\mathscr{G}') = \{\, P \in \mathcal{O}^{\mathrm{lin}}(\mathscr{G}) \mid P \text{ is linear} \,\}$.

Let $M \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G})$ be such that $|\mathscr{L}(M)|_{\beta\eta}$ is linear. By Lemma 3.5, there is $Q \in \Pi(\mathscr{L}(M))$ of type $\mathscr{L}(s)$. By applying Lemma 3.7 to M and Q, we have $N \in \Lambda^{\mathrm{lin}}(\Sigma_0')$ with $\tau_0'(N) = [s, \mathscr{L}(s)]$ and $\mathscr{L}'(N) \equiv (Q)^\dagger$. We have $N \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G}')$. Together with Lemma 3.4, it holds that $\mathscr{L}'(N) \equiv (Q)^\dagger =_\beta \widetilde{Q} \equiv \mathscr{L}(M)$.

Let $N \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G}')$. By applying Lemma 3.7 to N, we have $M \in \Lambda^{\mathrm{lin}}(\Sigma_0)$ and $Q \in \Pi(\mathscr{L}(M)) - \Lambda^{\mathrm{aff}}(\overline{\Sigma_1})$ such that $\tau_0(M) = s$, $\tau_1'(Q) = \mathscr{L}(s)$, and $\mathscr{L}'(N) \equiv (Q)^\dagger$. We have $M \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G})$. Together with Lemma 3.4, it holds that $\mathscr{L}'(N) \equiv (Q)^\dagger =_\beta \widetilde{Q} \equiv \mathscr{L}(M)$. □

De Groote and Pogodalla [18, 19] 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 3.9. Let a non-duplicating CFTG G consist of the following productions:

$$S \to P\mathsf{ab}, \quad P[x_1, x_2] \to P(\mathsf{c}x_1)(\mathsf{c}S) \mid \mathsf{d}x_1x_2,$$

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	$\lambda x_1^o x_2^o.\mathsf{d}^{o^2 \to o} x_1 x_2$

When we apply the linearization method given in Definition 3.6 to $\mathscr{G}$, we get the following linear ACG $\mathscr{G}^l$ whose distinguished type is $[s, o]$:

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

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 [10], which covers the *monadic* case only.

Corollary 3.10. *For every non-duplicating CFTG G, there is a linear CFTG G' such that G and G' generate the same tree language.*

De Groote and Pogodalla [19]'s encoding method of LCFRSs by linear ACGs described in Section 2.2.1 can be applied to MCFGs also. 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, while $\mathscr{G}'$ is not. We here explain that lexical entries of $\mathscr{G}'$ that cannot be interpreted as the encodings of productions of an LCFRS are useless. For instance, for the production rule $S \to f()$ with $f() = \langle\mathsf{a}\rangle$ of an MCFG, $\mathscr{G}$ has the corresponding abstract constant A of type S that is mapped to the term

$$\lambda z^{str\to str}.z\mathsf{a}.$$

The linearization method in Definition 3.6 lets $\mathscr{G}'$ have the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0'$	$\tau_0'(\mathsf{x})$	$\mathscr{L}'(\mathsf{x})$
$[\![\mathsf{A}, \lambda z^{\mathit{str}\to\mathit{str}}.z\mathsf{a}]\!]$	$[S, (\mathit{str} \to \mathit{str}) \to \mathit{str}]$	$\lambda z^{\mathit{str}\to\mathit{str}}.z\mathsf{a}$
$[\![\mathsf{A}, \lambda z^{\mathit{str}\to\bar{o}\to o}.z\mathsf{a}]\!]$	$[S, (\mathit{str} \to \bar{o} \to o) \to \bar{o} \to o]$	$\lambda z^{\mathit{str}\to o}.z\mathsf{a}$
$[\![\mathsf{A}, \lambda z^{\overline{\mathit{str}}\to\mathit{str}}.z\bar{\mathsf{a}}]\!]$	$[S, (\overline{\mathit{str}} \to \mathit{str}) \to \mathit{str}]$	$\lambda z^{\mathit{str}}.z$
$[\![\mathsf{A}, \lambda z^{\overline{\mathit{str}}\to\bar{o}\to o}.z\bar{\mathsf{a}}]\!]$	$[S, (\overline{\mathit{str}} \to \bar{o} \to o) \to \bar{o} \to o]$	$\lambda z^{o}.z$

Recall that a nonterminal symbol of rank k in an LCFRS is encoded as an abstract atomic type mapped to $(\mathit{str}^k \to \mathit{str}) \to \mathit{str}$ by the lexicon in the corresponding linear ACG. In the above table, the new atomic types $[S, (\mathit{str} \to \bar{o} \to o) \to \bar{o} \to o]$ and $[S, (\overline{\mathit{str}} \to \bar{o} \to o) \to \bar{o} \to o]$ are, however, mapped to $(\mathit{str} \to o) \to o$ and $o \to o$, respectively. Those atomic types cannot be interpreted as the encodings of nonterminals of an LCFRS. Consequently, the abstract constants $[\![\mathsf{A}, \lambda z^{\mathit{str}\to\bar{o}\to o}.z\mathsf{a}]\!]$ and $[\![\mathsf{A}, \lambda z^{\overline{\mathit{str}}\to\bar{o}\to o}.z\bar{\mathsf{a}}]\!]$ cannot be interpreted as the encodings of some productions of an LCFRS. We show that indeed those abstract constants are useless.

Let

$$\Gamma = \{\, (\alpha_1 \to \cdots \to \alpha_k \to \mathit{str}) \to \mathit{str} \mid \alpha_i \in \{\mathit{str}, \overline{\mathit{str}}\} \,\}.$$

We prove that for every new abstract atomic type $[p, \alpha]$ with $p \neq s'$, where s' is the distinguished type of the affine ACG $\mathscr{G}$, if $\alpha \notin \Gamma$, then $[p, \alpha]$ is useless in $\mathscr{G}'$. Let

$$\Delta = \{\, (\alpha_1 \to \cdots \to \alpha_k \to \bar{o} \to o) \to \bar{o} \to o \mid \alpha_i \in \{\mathit{str}, \overline{\mathit{str}}\} \,\}.$$

First we show that if a variable-free term $N \in \Lambda^{\mathrm{lin}}(\Sigma_0')$ has an atomic type $[p, \gamma]$ with $p \neq s'$, then $\gamma \in \Gamma \cup \Delta$ holds, and moreover, if a subterm N' of N has a type $[q, \gamma']$, then either $\gamma, \gamma' \in \Gamma$ or $\gamma, \gamma' \in \Delta$. Let $[\![\mathsf{A}, Q]\!]$ be the head of N, $\tau_0(\mathsf{A}) = p_1 \to \cdots \to p_m \to q$, $\tau_1'(Q) = \alpha_1 \to \cdots \to \alpha_m \to \beta$. By $\tau_0'([\![\mathsf{A}, Q]\!]) = ([p_1, \alpha_1] \to \cdots \to [p_m, \alpha_m] \to [q, \beta])^{\ddagger}$, N has the form

$$N \equiv [\![\mathsf{A}, Q]\!] N_{i_1} \dots N_{i_n}$$

where $\{i_1, \dots, i_n\} = \{\, i \mid \alpha_i \notin \mathcal{T}(\overline{\mathscr{A}_1}) \,\}$. Let

$$Q \equiv \lambda y_1^{\alpha_1} \dots y_m^{\alpha_m} z^{\beta_0}.y_1(\lambda \vec{x}_1. \dots y_m(\lambda \vec{x}_m.zQ_1 \dots Q_k) \dots)$$

where $\beta = \beta_0 \to \gamma$ for $\gamma \in \{\mathit{str}, \bar{o} \to o\}$. Then, α_1 has the form $\alpha_1' \to \gamma \notin \mathcal{T}(\overline{\mathscr{A}_1})$. Thus, $i_1 = 1$. By applying the induction hypothesis to N_1, we see $\alpha_1 \in \Gamma \cup \Delta$ and $\alpha_1 = (\tau_1'(\vec{x}_1) \to \gamma) \to \gamma$. Applying the same discussion to $\alpha_2, \dots, \alpha_m$ repeatedly, we have $n = m$, $i_j = j$, and $\alpha_i = (\tau_1'(\vec{x}_i) \to \gamma) \to \gamma \in \Gamma \cup \Delta$. $\alpha_i \in \Gamma \cup \Delta$ implies that the type of each variable in $\vec{x}_i$ is either str or $\overline{\mathit{str}}$. Thus, Q_j consists of atomic terms of types in $\{\mathit{str}, \overline{\mathit{str}}\}$ and hence the type of each Q_j is either str or $\overline{\mathit{str}}$. Therefore,

$$\beta = \beta_0 \to \gamma = (\tau_1'(Q_1) \to \cdots \to \tau_1'(Q_m) \to \gamma) \to \gamma \in \Gamma \cup \Delta.$$

Let $N \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G}')$. N has the form $[\![\mathsf{S}, Q]\!]N'$, where $\tau_0'([\![\mathsf{S}, Q]\!]) = [S, \alpha] \to [s', \mathit{str}]$. By applying the above claim to N', we have $\alpha \in \Gamma \cup \Delta$. Let

$$Q \equiv \lambda y^{(\alpha_1 \to \alpha_2) \to \alpha_3}.y(\lambda x^{\alpha_4}.x) \in \Pi(\mathscr{L}(\mathsf{S})) = \Pi(\lambda y^{(\mathit{str} \to \mathit{str}) \to \mathit{str}}.y(\lambda x^{\mathit{str}}.x)),$$

where $\alpha = (\alpha_1 \to \alpha_2) \to \alpha_3$. We have $\alpha_3 = \mathit{str}$ by $\tau_1'(Q) = \alpha \to \mathit{str}$, $\alpha_1 \to \alpha_2 = \alpha_4 \to \alpha_4$ by the well-typedness of Q, and $\alpha_2 = \alpha_3$ by $\alpha \in \Gamma \cup \Delta$. Therefore, $Q \equiv \mathscr{L}(\mathsf{S})$ and $\alpha = (\mathit{str} \to \mathit{str}) \to \mathit{str} \in \Gamma$. The above claim entails that if a subterm of $N \in \mathcal{A}(\mathscr{G}')$ has a type $[q, \gamma]$ with $q \neq s'$, then $\gamma \in \Gamma$. Therefore, every useful abstract atomic type of $\mathscr{G}'$ is mapped to $(\mathit{str}^k \to \mathit{str}) \to \mathit{str}$ for some non-negative integer k, which is interpreted as the encoding of a nonterminal of rank k in an LCFRS.

Note that the LCFRS obtained from an MCFG through our linearization method may have nonterminals of rank 0, though usual definitions of an LCFRS do not allow nonterminals to have rank 0. Mathematically speaking, allowing or disallowing nonterminals of rank 0 does not matter at all. The reason why usual definitions of an LCFRS do not allow nonterminals to have rank 0 is just to avoid redundancy. If a nonterminal A_k has rank 0, and a production $A_k \to f(B_1, \ldots, B_m)$ is in an LCFRS, then f is the function that maps the empty sequence to the empty sequence and each B_i has rank 0. If A_k appears in the right-hand side of a production as $C \to g(A_1, \ldots, A_k, \ldots, A_n)$, we can eliminate A_k from the right-hand side without modifying the function g, which is a function from $\prod_{1 \leq i \leq n} (T^*)^{\rho(A_i)} = \prod_{\substack{1 \leq i \leq n \\ i \neq k}} (T^*)^{\rho(A_i)}$ to $(T^*)^{\rho(C)}$. Note that here we work modulo the associativity of the cartesian product, i.e., we identify $(T^*)^n \times (T^*)^m$ with $(T^*)^{n+m}$. Thus, we can simply ignore nonterminals of rank 0 if they occur in an LCFRS.

Therefore, our result covers the following theorem shown by Seki et al. [49]

Corollary 3.11. *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 3.6 itself does not depend on the order of the given affine ACG except for checking whether useless abstract atomic types or useless atomic constants exist. For the general case, however, the linearized ACG given in Definition 3.6 may generate a strictly larger language than the original affine ACG. In the remainder of this chapter, we present a linearization method for general affine ACGs.

Example 3.12. 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	$p \to q \to q$	$\lambda y^o z^o.\mathsf{b}z$
C	$q \to s$	$\lambda z^o.z$
D	$(p \to s) \to s$	$\lambda x^{o\to o}.\mathsf{a}(x\mathsf{e})$

We have

$$\mathcal{A}^{\mathrm{lin}}(\mathscr{G}) = \{\, \mathsf{D}(\lambda y_1^p.\mathsf{D}(\lambda y_2^p.\ldots\mathsf{D}(\lambda y_n^p.\mathsf{C}(\mathsf{B}y_{i_1}(\mathsf{B}y_{i_2}(\ldots(\mathsf{B}y_{i_n}\mathsf{A})\ldots))))\ldots)) \mid n \geq 0,\ \{i_1,\ldots,i_n\} = \{1,\ldots,n\}\,\}$$

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

$\mathscr{G}'$ consists of the following lexical entries:

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

The last lexical entry is useless. We have

$$\mathcal{A}^{\mathrm{lin}}(\mathscr{G}') = \{\, \overbrace{\mathsf{D}'(\ldots(\mathsf{D}'}^{m\text{-times}}(\mathsf{C}'(\overbrace{\mathsf{B}'(\ldots(\mathsf{B}'}^{n\text{-times}}\mathsf{A}')\ldots))\ldots) \mid m,n \geq 0\,\}$$

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

where A', B', C', D' are, respectively, the first, second, third, fourth abstract constants of $\mathscr{G}'$ shown in the above table. 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 $\mathcal{A}^{\mathrm{lin}}(\mathscr{G})$.

Our alternative linearization method presented later gives the linear ACG $\mathscr{G}''$ consisting of the following lexical entries (useless lexical entries are sup-

pressed):

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

where $[p, \overline{o}]$ is mapped to $o \to o$. For

$$M \equiv \mathsf{D}(\lambda y^p.\mathsf{C}(\mathsf{B}y\mathsf{A})) \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G}),$$

we have

$$N \equiv \mathsf{D}''(\lambda y^{[p,\overline{o}]}.\mathsf{C}''(\mathsf{B}''y\mathsf{A}'')) \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G}'')$$

where A'', B'', C'', D'' are, respectively, the first, second, third, fourth abstract constants of $\mathscr{G}''$ shown in the above table. It is easy to check that $\mathscr{L}(M) = \mathscr{L}''(N) = \mathsf{a}(\mathsf{b}\#)$. We have $\mathcal{O}(\mathscr{G}) = \mathcal{O}(\mathscr{G}'')$.

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 Σ_0'' is defined by

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

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

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

$\mathscr{L}_0$ is relabeling. We have $\widetilde{\mathscr{L}_1(N)} \equiv \mathscr{L} \circ \mathscr{L}_0(N)$ for $N \in \Lambda^{\mathrm{lin}}(\Sigma_0'')$.

Lemma 3.13. *For every $Q \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$ and $\alpha \in \mathcal{T}(\mathscr{A}_0)$, the following statements are equivalent:*

(i) There is $M \in \Lambda^{\mathrm{lin}}(\Sigma_0)$ of type α such that $\mathscr{L}(M) \equiv \widetilde{Q}$.

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

Proof. The direction [(ii) $\Rightarrow$ (i)] can be shown by letting $M \equiv \mathscr{L}_0(N)$.

The opposite direction [(i) $\Rightarrow$ (ii)] can be shown by induction on M. Let $N \equiv \chi(M, Q)$ where

- $\chi(\mathsf{a}, Q) \equiv [\![\mathsf{a}, Q]\!]$,
- $\chi(x^\alpha, x^\beta) \equiv x^{[\alpha,\beta]}$,
- $\chi(M_1 M_2, Q_1 Q_2) \equiv \chi(M_1, Q_1)\chi(M_2, Q_2)$,
- $\chi(\lambda x^\alpha.M', \lambda x^\beta.Q') \equiv \lambda x^{[\alpha,\beta]}.\chi(M', Q')$.

Here, $\mathscr{L}_0(\chi(M,Q)) \equiv M$, $\mathscr{L}_1(\chi(M,Q)) \equiv Q$, and $\chi(\mathscr{L}_0(N), \mathscr{L}_1(N)) \equiv N$ hold. □

Lemmas 3.5 and 3.13 imply

$$\{\, M \in \mathcal{A}(\mathscr{G}) \mid |\mathscr{L}(M)|_\beta \text{ is linear} \,\} = \{\, \mathscr{L}_0(N) \mid N \in \mathcal{A}(\mathscr{G}'') \,\}. \tag{3.5}$$

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 3.4, it is enough to define a new lexicon $\mathscr{L}''$ so that

$$\mathscr{L}''(N) =_{\beta\eta} (\mathscr{L}_1(N))^\dagger \tag{3.6}$$

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

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

Here we identify σ with its homomorphic extension. For each $[\alpha,\beta] \in \mathcal{T}(\mathscr{A}_0'')$ such that $\beta \in \widehat{\mathcal{T}}(\{o\}) - \mathcal{T}(\{\bar{o}\})$, 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 $\beta_i \notin \mathcal{T}(\{\bar{o}\})$ iff $i \in I$. Let $I = \{i_1,\ldots,i_k\}$ with $i_j < i_{j+1}$ for each $1 \le j < k$ and $J = \{j_1,\ldots,j_l\}$. If $i \in J$, then the linear combinators $Z^{\sigma([\alpha_i,\beta_i])}$ and $Z^{\sigma([\alpha_i,\beta_i])\to o\to o}$ (given in Lemma 2.12) are well-defined by the definition of σ. We define

$$X_\alpha^\beta \equiv \lambda y^{\sigma([\alpha,\beta])} x_{i_1}^{(\beta_{i_1})^\dagger} \ldots x_{i_k}^{(\beta_{i_k})^\dagger} . y^{\sigma([\alpha,\beta])} R_1 \ldots R_m$$

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

and

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

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

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

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

Now, we give a new linearization method as follows.

Definition 3.14. 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

$$\theta(\llbracket \mathsf{a}, Q \rrbracket) \equiv \begin{cases} |Y_{\tau_0(\mathsf{a})}^{\tau_1'(Q)}(Q)^\dagger|_\beta & \text{if } Q \notin \Lambda^{\mathrm{aff}}(\overline{\Sigma_1}), \\ Z^{\sigma(\tau_0''(\llbracket \mathsf{a},Q \rrbracket))} & \text{if } Q \in \Lambda^{\mathrm{aff}}(\overline{\Sigma_1}). \end{cases}$$

Thus, if the given affine ACG $\mathscr{G}$ belongs to $\mathbf{G}^{\mathrm{aff}}(m,n)$, then $\mathscr{G}''$ belongs to $\mathbf{G}^{\mathrm{lin}}(m, \max\{2,n\})$.

Lemma 3.15. *Given $N \in \Lambda(\Sigma_0'')$ of type $[\alpha,\beta]$ such that $\beta \notin \mathcal{T}(\{\bar{o}\})$ and $\mathscr{L}_1(N) \in \widehat{\Lambda}^{\mathrm{aff}}(\Sigma_1)$, we have*

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

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

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

Proof. We assume that N is in long normal form for simplicity. The proof is done by induction on N.

Case 1. Suppose that $N = \lambda x_1^{[\alpha_1,\beta_1]} \dots x_m^{[\alpha_m,\beta_m]}.N'$ have type $[\alpha,\beta] = [\alpha_1,\beta_1] \to \dots \to [\alpha_m,\beta_m] \to [p,\beta_0]$ with $m \geq 1$. Let us partition the set $\{1,\dots,m\}$ into two subsets I and J so that $\beta_i \notin \mathcal{T}(\{\bar{o}\})$ iff $i \in I$ and $\beta_i \in \mathcal{T}(\{\bar{o}\})$ iff $i \in J$. Let the elements of I be $i_1,\dots,i_k$ with $i_j < i_{j+1}$ for $1 \leq j < k$. Since $\mathrm{Fv}(N') = \mathrm{Fv}(N) \cup \{x_1,\dots,x_m\}$, we have

$$\phi_{N'} = \phi_N \cup \{\, x_i^{\sigma([\alpha_i,\beta_i])} \mapsto R_i \mid 1 \leq i \leq m \,\}$$
$$\text{where } R_i = \begin{cases} Y_{\alpha_i}^{\beta_i} x_i & \text{if } i \in I, \\ Z^{\sigma([\alpha_i,\beta_i])} & \text{if } i \in J. \end{cases}$$

Thus, together with the induction hypothesis and $X_p^{\beta_0]} = \lambda z^{(\beta_0)^\dagger}.z$,

$$\begin{aligned} (\mathscr{L}_1(N))^\dagger &\equiv \lambda x_{i_1}^{(\beta_{i_1})^\dagger} \dots x_{i_k}^{(\beta_{i_k})^\dagger}.(\mathscr{L}_1(N'))^\dagger \\ &=_{\beta\eta} \lambda x_{i_1}^{(\beta_{i_1})^\dagger} \dots x_{i_k}^{(\beta_{i_k})^\dagger}.\big(X_p^{\beta_0}\mathscr{L}''(N')\phi_{N'}\big) \\ &\twoheadrightarrow_\beta \lambda x_{i_1}^{(\beta_{i_1})^\dagger} \dots x_{i_k}^{(\beta_{i_k})^\dagger}.\big(\mathscr{L}''(N')[R_i/x_i^{\sigma([\alpha_i,\beta_i])}]_{1\leq i\leq m}\big)\phi_N. \end{aligned} \tag{3.7}$$

On the other hand,

$$\begin{aligned}
&X^{\beta}_{\alpha}\mathscr{L}''(N)\phi_N\\
&\equiv \big(\lambda y^{\sigma([\alpha,\beta])}x_{i_1}^{(\beta_{i_1})^\dagger}\dots x_{i_k}^{(\beta_{i_k})^\dagger}.yR_1\dots R_m\big)\mathscr{L}''(\lambda x_1^{[\alpha_1,\beta_1]}\dots x_m^{[\alpha_m,\beta_m]}.N')\phi_N\\
&\twoheadrightarrow_\beta \lambda x_{i_1}^{(\beta_{i_1})^\dagger}\dots x_{i_k}^{(\beta_{i_k})^\dagger}.\mathscr{L}''(\lambda x_1^{[\alpha_1,\beta_1]}\dots x_m^{[\alpha_m,\beta_m]}.N')R_1\dots R_m\phi_N\\
&\twoheadrightarrow_\beta \lambda x_{i_1}^{(\beta_{i_1})^\dagger}\dots x_{i_k}^{(\beta_{i_k})^\dagger}.\big(\mathscr{L}''(N')[R_i/x_i^{\sigma([\alpha_i,\beta_i])}]_{1\le i\le m}\big)\phi_N. \qquad (3.8)
\end{aligned}$$

(3.7) and (3.8) coincide.

Case 2. Suppose that $N = x^{[\alpha,\beta]}N_1\dots N_m$ where $\alpha = \alpha_1 \to \dots \to \alpha_m \to p$, $\beta = \beta_1 \to \dots \to \beta_m \to \beta_0$. Let us partition the set $\{1,\dots,m\}$ into two subsets I and J so that $\beta_i \notin \mathcal{T}(\{\bar{o}\})$ iff $i \in I$ and $\beta_i \in \mathcal{T}(\{\bar{o}\})$ iff $i \in J$. Let the elements of I be $i_1,\dots,i_k$ with $i_j < i_{j+1}$ for $1 \le j < k$ and $J = \{j_1,\dots,j_l\}$. By the induction hypothesis,

$$\begin{aligned}
(\mathscr{L}_1(N))^\dagger &\equiv x^{(\beta)^\dagger}(\mathscr{L}_1(N_{i_1}))^\dagger\dots(\mathscr{L}_1(N_{i_k}))^\dagger\\
&=_{\beta\eta} x^{(\beta)^\dagger}(X_{i_1}\mathscr{L}''(N_{i_1})\phi_{N_{i_1}})\dots(X_{i_k}\mathscr{L}''(N_{i_k})\phi_{N_{i_k}}) \qquad (3.9)
\end{aligned}$$

where $X_i = X^{\beta_i}_{\alpha_i}$.

On the other hand, since $X_p^{\beta_0}$ is the identity,

$$\begin{aligned}
&X_p^{\beta_0}\mathscr{L}''(N)\phi_N \twoheadrightarrow_\beta \mathscr{L}''(N)\phi_N\\
&\quad\equiv \big(Y^{\beta}_{\alpha}x^{(\beta)^\dagger}\big)\big(\mathscr{L}''(N_1)\phi_{N_1}\big)\dots\big(\mathscr{L}''(N_m)\phi_{N_m}\big)\\
&\quad\twoheadrightarrow_\beta \big(\lambda y_1^{\sigma([\alpha_1,\beta_1])}\dots y_m^{\sigma([\alpha_m,\beta_m])}\vec{z}.M_{j_1}(\dots(M_{j_l}(x^{(\beta)^\dagger}L_{i_1}\dots L_{i_k}\vec{z}))\dots)\big)\\
&\qquad\qquad\big(\mathscr{L}''(N_1)\phi_{N_1}\big)\dots\big(\mathscr{L}''(N_m)\phi_{N_m}\big)
\end{aligned}$$

where

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

For $j \in J$, it is easy to see that if a subterm (a free variable in particular) of N_j has a type $[\gamma,\delta]$, then $\delta = \overline{\mathscr{L}(\gamma)} \in \mathcal{T}(\{\bar{o}\})$. Thus, ϕ_{N_j} substitutes linear combinators for all the free variables of $\mathscr{L}''(N_j)$. Therefore, $\mathscr{L}''(N_j)\phi_{N_j}$ and $M_j[\mathscr{L}''(N_j)\phi_{N_j}/y_j]$ are linear combinators. By the type $o \to o$ of $M_j[\mathscr{L}''(N_j)\phi_{N_j}/y_j]$, we see $M_j[\mathscr{L}''(N_j)\phi_{N_j}/y_j] \twoheadrightarrow_\beta \lambda z^o.z$. Therefore

$$\begin{aligned}
\mathscr{L}''(N)\phi_N &\twoheadrightarrow_\beta \lambda\vec{z}.x^{(\beta)^\dagger}(X_{i_1}\mathscr{L}''(N_{i_1})\phi_{N_{i_1}})\dots(X_{i_k}\mathscr{L}''(N_{i_k})\phi_{N_{i_k}})\vec{z}\\
&\twoheadrightarrow_\eta x^{(\beta)^\dagger}(X_{i_1}\mathscr{L}''(N_{i_1})\phi_{N_{i_1}})\dots(X_{i_k}\mathscr{L}''(N_{i_k})\phi_{N_{i_k}}). \qquad (3.10)
\end{aligned}$$

(3.9) and (3.10) coincide.

Case 3. Suppose that $N = [\![\mathsf{a}, Q]\!] N_1 \dots N_m$. This case can be treated by replacing the head variable $x^{[\alpha,\beta]}$ in the previous case with $[\![\mathsf{a}, Q]\!]$. □

Theorem 3.16. *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 $\mathcal{O}^{\mathrm{lin}}(\mathscr{G}'') = \{\, P \in \mathcal{O}^{\mathrm{lin}}(\mathscr{G}) \mid P \text{ is linear} \,\}$.*

Proof. Applying Lemma 3.15 to $N \in \mathcal{A}^{\mathrm{lin}}(\mathscr{G}'')$, we get the equation (3.6), since ϕ_N is the empty substitution and $X_s^{\mathscr{L}(s)}$ is the identity. □

3.4.3 Open Issues

In this section, we have presented how vacuous lambda abstraction can be eliminated from affine ACGs. It is natural to ask whether or not we can transform a λK-ACG $\mathscr{G}$ into a λI-ACGs $\mathscr{G}'$ such that $\mathcal{O}^{\mathrm{lin}}(\mathscr{G}') = \{\, P \in \mathcal{O}^{\mathrm{lin}}(\mathscr{G}') \mid P \text{ is } \lambda\mathrm{I} \,\}$. That is future work. Recall that the affine term $\lambda x^{(o\to o\to o)\to o}.x(\lambda z_1^o z_2^o.z_1)$ is retyped as $\lambda x^{(o\to\bar{o}\to o)\to o}.x(\lambda z_1^o z_2^{\bar{o}}.z_1)$ by our method. However, we cannot retype the non-affine term

$$\lambda x^{(o\to o)\to o}.\mathsf{f}\big(x^{(o\to o)\to o}(\lambda z^o.\mathsf{a})\big)\big(x^{(o\to o)\to o}(\lambda z^o.z^o)\big)$$

as

$$\lambda x^{(?\to o)\to o}.\mathsf{f}\big(x^{(\bar{o}\to o)\to o}(\lambda z^{\bar{o}}.\mathsf{a})\big)\big(x^{(o\to o)\to o}(\lambda z^o.z^o)\big).$$

The author conjectures that one can eliminate vacuous λ-abstraction from *semi-affine ACGs*, where a λK-term is *semi-affine* if for every free variable x of any subterm, either

- x occurs exactly once, or
- x has an atomic or second-order type.

For instance, for the semi-affine term

$$\big(\lambda u^{o\to o} v^{o\to o} z^o.\mathsf{f}(u^{o\to o}z^o)(v^{o\to o}z^o)\big)(\lambda x^o.x^o)(\lambda y^o.\mathsf{a}^o),$$

although

$$\big(\lambda u^{o\to o} v^{\bar{o}\to o} z^{?}.\mathsf{f}(u^{o\to o}z^o)(v^{\bar{o}\to o}z^{\bar{o}})\big)(\lambda x^o.x^o)(\lambda y^{\bar{o}}.\mathsf{a}^o)$$

is not well-typed, the term

$$\big(\lambda u^{o\to o} v^o z^o.\mathsf{f}(u^{o\to o}z^o)v^o\big)(\lambda x^o.x^o)\mathsf{a}^o$$

obtained by eliminating barred terms and types from the previous one is well-typed. If the construction based on this idea is correct, it implies that every

CFTG has a corresponding non-deleting CFTG whose IO-tree language is equivalent, because every CFTG can be encoded with respect to the IO-tree language by an ACG belonging to $\mathbf{G}^{\mathrm{K}}_{\mathrm{tree}}(2,2)$ such that only variables of atomic types occur more than once in its lexical entries, and vice versa. This also entails Fisher's result [8, 9] that every CFTG has a corresponding non-deleting CFTG whose IO-*string* language is equivalent.

On the other hand, the author conjectures that it is impossible to linearize λI-ACGs, i.e., there is a λI-ACG $\mathscr{G}$ such that for any linear ACG $\mathscr{G}'$,

$$\{\, P \in \mathcal{O}^{\mathrm{lin}}(\mathscr{G}) \mid P \text{ is linear} \,\} \neq \mathcal{O}^{\mathrm{lin}}(\mathscr{G}').$$

However, it seems difficult to prove it, as we have not yet found any recursively enumerable language that cannot be generated by any linear ACG. At least, we know that the above statement is true when we consider only second-order ACGs. The following second-order λI-ACG $\mathscr{G} \in \mathbf{G}^{\mathrm{I}}_{\mathrm{string}}(2,2)$ generates a non-semilinear language, which cannot be generated by any second-order linear ACG by Proposition 2.20.

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}_1(\mathsf{x})$
A	s	$/\mathsf{c}/$
B	$s \to s$	$\lambda x^{str}.x + x$

$$\mathcal{O}^{\mathrm{lin}}(\mathscr{G}) = \{\, /\mathsf{c}^{2^n}/ \mid n \in \mathbb{N} \,\}.$$

3.5 Summary

This chapter has discussed non-linear extensions of ACGs. The original ACGs have two kinds of linearity constraint: one on the lexicons, one on the abstract languages. Section 3.3 shows that relaxing the linearity constraint on the abstract language does not enlarge the class of ACLs as much as allowing non-linear lexicons.

The main issue of this chapter is comparison of generative capacity of original linear ACGs and affine ACGs, which may have vacuous lambda abstraction. In Section 3.4, we have established the equivalence between affine ACGs and linear ACGs by presenting two procedures transforming affine ACGs into linear ACGs. The first one is for second-order affine ACGs and the second one, which is an elaboration of the first one, is for third or higher-order ACGs.

The first linearization method covers the results obtained by Seki et al. [49] and Fujiyoshi [10]. Seki et al. have shown the equivalence between MCFGs and LCFRSs, and Fujiyoshi has shown the equivalence between

monadic non-duplicating CFTG and monadic linear CFTG with respect to tree languages. Their and our common basic idea is to eliminate deleting operations and some strings or subtrees in productions from the given grammar that would be erased by deleting operations during derivations. Their techniques create new productions from an original production by replacing nonterminals by new nonterminals of less ranks and fewer arguments. In this sense, their techniques work only on "second-order" settings. On the other hand, our linearization method generalizes their techniques for higher-order settings. From a type appearing in the given affine ACG, our method constructs types by eliminating subtypes of the original.

Moreover, our method entails the new result as a corollary that non-duplicating CFTGs and linear CFTGs are equivalent with respect to tree languages. This demonstrates that studying ACGs would be investigation of mildly context-sensitive grammars. Our result strengthens de Groote and Pogodalla's view that ACGs can be the kernel of a grammatical framework; not only well known grammar formalisms themselves, but also a transformation of existing grammar formalisms are encoded in ACGs.

The second linearization method can also be applied to second-order affine ACGs actually and it transforms non-duplicating CFTGs into linear CFTGs, but it does not give a conversion from MCFGs to LCFRS. The advantage of the second method is not only the fact that it is applicable to every affine ACGs, but also it preserves the original derivation structures in the sense that the original abstract language is the image of the new abstract language under a relabeling lexicon, namely $\mathscr{L}_0$ on page 40. As we will see in Section 5.6, due to this property, the second linearization method is still valid for multi-dimensional extensions of affine ACGs.

Though every affine ACG admits linearization, our both constructions increase the size of the given grammar exponentially due to the definition of Π, so there still exists 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 (not counting other useless types) which correspond to noun phrases as third person singular subjects, plural subjects, and objects respectively.

Chapter 4

Lexicalized Abstract Categorial Grammars

4.1 Introduction

A formalization of the notion of *lexicalized grammars* is given by Schabes et al. [47, 48] A grammar is called lexicalized if each of its lexical entries that contribute to deriving an element of the language contains an item, called a *lexical item*, that explicitly appears on the surface of the derived structure. From the point of lexicalists' view, which thinks that linguistic phenomena should be accounted for by the inherent information in the lexical entries, to be lexicalized is a natural requirement. Moreover, not only from the lexicalists' point of view, but also from the point of view of the computational complexity, to be lexicalized is often thought to be desirable. At least, the decidability of the universal membership problem for lexicalized grammar is guaranteed in general.

Preceding research showed that some grammar formalisms admit lexicalization. Greibach [13] showed that every CFG has an equivalent (modulo the empty string) CFG in Greibach normal form, which can be thought of as a lexicalized grammar, because every production contains a terminal symbol in its right hand side. Schabes et al. [47, 48] presented a lexicalization method that converts finitely ambiguous TAGs into lexicalized TAGs.

This chapter is devoted to discussion of lexicalized ACGs. We say that an ACG is *lexicalized* iff every abstract constant is mapped to an object term containing an object constant. Although the origin of ACGs is located in a history of categorial grammars, ACGs are not necessarily lexicalized grammars by definition unlike usual categorial grammars.

In Section 4.2, which is based on Yoshinaka and Kanazawa's paper [61],

the generative capacity and complexity of lexicalized ACGs is discussed. We see that the universal membership problem for lexicalized ACG is NP-complete. This result contrasts with the one for general ACGs and suggests that the restriction to lexicalized ACGs may be preferable from the point of view of computational complexity as well as of lexicalism.

Although the paper [61] claims that every second-order ACG admits lexicalization, the proof is in error. Kanazawa and Yoshinaka [27] give a correct proof. The main result of this chapter is a generalization of that claim. In Section 4.3, we show that *semilexicalized ACGs*, which form a superclass of second-order ACGs, admit lexicalization.

4.2 Lexicalized ACGs

Definition 4.1. For an ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$, an abstract constant $\mathsf{a} \in \mathscr{C}_0$ is said to be *lexical with respect to* $\mathscr{L}$ if $\mathscr{L}(\mathsf{a})$ contains a constant; otherwise, i.e., if $\mathscr{L}(\mathsf{a})$ is a combinator, a is *nonlexical with respect to* $\mathscr{L}$. If every abstract constant of an ACG is lexical, it is called a *lexicalized ACG* and the class of lexicalized ACGs is denoted by $\mathbf{G}^{\mathrm{lex}}$.

If the lexicon $\mathscr{L}$ is clear from the context, we suppress the words "with respect to $\mathscr{L}$".

Note that there is a gap between the notion of a lexicalized ACG in this thesis and the definition of a lexicalized TAG by Schabes et al. [47,48] While labels of internal nodes in a tree are not regarded as lexical items in Schabes et al.'s setting, labels of both internal nodes and leaf nodes, which are represented with object constants in ACGs, are regarded as lexical items by our definition. Every ACG encoding a TAG is lexicalized in our setting. This gap comes from the high degree of abstractness and generality of ACGs; they generate not only trees, but also other various types of data. Although it might be possible to define the notion of "lexicalized ACGs" so that it comprehends the notion of lexicalized TAGs by Schabes et al., such a definition would be reasonable only for tree ACGs. For the sake of generality, we adopt the above definition for the term "lexicalized ACGs".

Lemma 4.2. *Let a type substitution* $\sigma : \{o\} \to \mathcal{T}(\{o\})$ *be defined as* $\sigma(o) =$ *str and a constant* c *have type str. For any* $\alpha \in \mathcal{T}(\{o\})$*, there is a closed term* $Z_{\mathsf{c}}^{\sigma(\alpha)}$ *of type* $\sigma(\alpha)$ *that contains exactly one occurrence of* c*.*

Proof. Let $Z_{\mathsf{c}}^{\sigma(\alpha)} = Z^{\mathit{str} \to \sigma(\alpha)}\mathsf{c}$ for the linear combinator $Z^{\mathit{str} \to \sigma(\alpha)}$ defined in Definition 2.12. □

Recall that the proofs of Proposition 2.15 and Corollary 2.19 use ACGs whose lexicons are just identities. Therefore, the results on the decidability of the non-emptiness problem and on the generation of a non-semilinear language hold for lexicalized ACGs. By Lemma 4.2, we can replace the ACG in the proof of Proposition 2.15 with a lexicalized string ACG, and similarly we see that there is a lexicalized string ACG in $\mathbf{G}^{\mathrm{lex}}_{\mathrm{string}}(3,2)$ that generates a non-semilinear string language. However, the complexity of the universal membership problem for ACGs is greatly improved in the case of lexicalized ACGs.

Proposition 4.3. *The universal membership problem for lexicalized ACGs is NP-complete.*

Proof. First we show that the problem is in NP. For a lexicalized ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s\rangle$, if $\mathscr{L}(M) \twoheadrightarrow_\beta P \in \Lambda(\Sigma_1)$, the number of occurrences of constants in M does not exceed the number of occurrences of constants in P. Let P have m occurrences of constants. $P \in \mathcal{O}(\mathscr{G})$ iff there are abstract constants $\mathsf{c}_1, \ldots, \mathsf{c}_n$ for some $n \leq m$ and a combinator X of type $\tau_0(\mathsf{c}_1) \to \cdots \to \tau_0(\mathsf{c}_n) \to s$ such that $\mathscr{L}(X\mathsf{c}_1 \ldots \mathsf{c}_n) \twoheadrightarrow_\beta P$. The size of a linear combinator X is bounded by a polynomial function of the size of its type and the number of β-reduction steps needed to eliminate redexes of a linear term is bounded by its size. This shows that the question "$P \in \mathcal{O}(\mathscr{G})$?" is in NP.

The NP-hardness can be derived from the NP-hardness of the implicational fragment of intuitionistic linear logic ($\mathbf{IMLL}^{-\circ}$) [28]. For a given sequent $\mathcal{S} : A_1, \ldots, A_n \Rightarrow B$ of $\mathbf{IMLL}^{-\circ}$, we define an ACG $\mathscr{G}^{\mathcal{S}} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s\rangle$ where $\Sigma_0 = \langle \mathscr{A}_{\mathcal{S}} \cup \{s\}, \{\mathsf{a}\}, \tau_0\rangle$, $\mathscr{A}_{\mathcal{S}}$ is the set of atomic formulas in the sequent $\mathcal{S}$, $s \notin \mathscr{A}_{\mathcal{S}}$, $\tau_0(\mathsf{a}) = (A_1 \to \cdots \to A_n \to B) \to s$, $\Sigma_1 = \langle \{o\}, \{\mathsf{c}\}, \{\mathsf{c} \mapsto \mathit{str}\}\rangle$, $\mathscr{L}(p) = \mathit{str}$ for all $p \in \mathscr{A}_{\mathcal{S}} \cup \{s\}$, and $\mathscr{L}(\mathsf{a}) = Z_{\mathsf{c}}^{\mathscr{L}(\tau_0(\mathsf{a}))}$, where $Z_{\mathsf{c}}^{\mathscr{L}(\tau_0(\mathsf{a}))}$ is defined in Lemma 4.2. Then, $\mathcal{S}$ is provable in $\mathbf{IMLL}^{-\circ}$ iff $\mathsf{c} \in \mathcal{O}(\mathscr{G}^{\mathcal{S}})$. □

Salvati [43] has given some general results on the generative capacity and complexity of lexicalized ACGs. Salvati has stated the above proposition in a more refined form.

Proposition 4.4 (Salvati [43]). *The universal membership problem for* $\mathbf{G}^{\mathrm{lex}}_{\mathrm{tree}}(2,2)$ *is NP-complete.*

A question that naturally suggests itself at this point is whether or not there is a lexicalized ACG generating an NP-complete language. He has also answered this question.[1]

[1] Yoshinaka and Kanazawa [61] have presented an example of a fourth-order lexicalized ACG generating an NP-complete language independently from Salvati.

Proposition 4.5 (Salvati [43]). *There is a lexicalized ACG in* $\mathbf{G}^{\mathrm{lex}}_{\mathrm{tree}}(3,1)$ *whose language is NP-complete.*

Kanazawa's theorem (Theorem 2.21) holds in a weaker form for lexicalized string ACGs. We say that a class $\mathbf{L}$ of string languages is an *AFL* if $\mathbf{L}$ is closed under union, concatenation, Kleene plus, ε-free homomorphism, inverse homomorphism, and intersection with regular sets.

Theorem 4.6 (Kanazawa [26]). *The class of string languages generated by ACGs in* $\mathbf{G}^{\cdot}\mathrm{lex}(m,n)$ *is a substitution-closed AFL for all* $m, n \geq 2$.

4.3 Lexicalization of Semilexicalized ACGs

Definition 4.7 (Salvati [43]). An ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$ is said to be *semilexicalized* iff for every abstract constant $\mathsf{a} \in \mathscr{C}_0$, either

- a is lexical, or
- $\tau_0(\mathsf{a})$ is second-order.

Every lexicalized ACG and second-order ACG is semilexicalized by definition.

Salvati's result on semilexicalized ACGs shows that the classes of semilexicalized ACGs and lexicalized ACGs have similar generative capacity and complexity.

Proposition 4.8 (Salvati [43]). *Every semilexicalized ACG generates a language in NP. The universal membership problem for semilexicalized ACGs is decidable.*

In this section, we show that *semilexicalized* ACGs admit lexicalization. Our goals are the following two theorems.

Theorem 4.9. *For every semilexicalized ACG* $\mathscr{G} \in \mathbf{G}(m,n)$ *with* $m \in \{2,3\}$, *there are lexicalized ACGs* $\mathscr{G}' \in \mathbf{G}(m, n+1)$ *and* $\mathscr{G}'' \in \mathbf{G}(m+1, n)$ *such that*

$$\mathcal{O}(\mathscr{G}') = \mathcal{O}(\mathscr{G}'') = \{\, R \in \mathcal{O}(\mathscr{G}) \mid R \text{ contains a constant} \,\}.$$

Theorem 4.10. *For every semilexicalized ACG* $\mathscr{G} \in \mathbf{G}(m,n)$ *with* $m \notin \{2,3\}$, *there is a lexicalized ACG* $\mathscr{G}' \in \mathbf{G}(m,n)$ *such that*

$$\mathcal{O}(\mathscr{G}') = \{\, R \in \mathcal{O}(\mathscr{G}) \mid R \text{ contains a constant} \,\}.$$

The above theorem for $m = 1$ is trivial. Each ACG in $\mathbf{G}(1, n)$ defines just a finite language. Lexicalization of a first-order ACG is just eliminating all the nonlexical constants from the abstract vocabulary.

Note that the notion of *lexicalization* in this thesis differs from Schabes et al.'s one [47, 48], which is concerned with grammars that generate strings through tree structures. They have defined the notion of lexicalization so that it preserves not only the string language but also the tree language yielding the string language. This definition requires grammars that should be lexicalized to be finitely ambiguous, i.e., there is no string in the language that has infinitely many trees yielding it in the tree language. Our lexicalization method converts any semilexicalized ACG, which may be infinitely ambiguous in the sense that there are infinitely many λ-terms in the abstract language that are mapped to the same object term modulo $\beta\eta$, into a lexicalized ACG, where the original abstract language may be lost.

4.3.1 Basic Idea

This subsection explains our basic strategy for lexicalizing semilexicalized ACGs.[2]

In the sequel, for an ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$, let us denote the set of lexical constants by $\mathscr{C}_0^+$ and the set of nonlexical constants by $\mathscr{C}_0^-$. We denote the lexical part of the abstract vocabulary by $\Sigma_0^+ = \langle \mathscr{A}_0, \mathscr{C}_0^+, \tau_0 \rangle$, and the nonlexical part by $\Sigma_0^- = \langle \mathscr{A}_0, \mathscr{C}_0^-, \tau_0 \rangle$. An ACG is semilexicalized iff Σ_0^- is second-order. For $M, N \in \Lambda(\Sigma_0)$, we write $M \overset{\mathscr{L}}{\approx} N$ (or $M \approx N$ if $\mathscr{L}$ is clear from the context) iff $\tau_0(M) = \tau_0(N)$ and $\mathscr{L}(M) = \mathscr{L}(N)$. For two sets Γ and Δ of terms, we write $\Gamma \approx \Delta$ iff for every $M \in \Gamma$, there is $N \in \Delta$ with $M \approx N$ and vice versa.

For eliminating nonlexical constants from a semilexicalized ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$, we construct a new ACG $\mathscr{G}' = \langle \Sigma_0', \Sigma_0, \mathscr{L}', s' \rangle$ such that

- for every $\mathsf{A} \in \mathscr{C}_0'$, $\mathscr{L}'(\mathsf{A})$ contains at least one lexical (w.r.t. $\mathscr{L}$) constant $\mathsf{a} \in \mathscr{C}_0^+$.
- $\mathcal{O}(\mathscr{G}') \overset{\mathscr{L}}{\approx} \mathcal{A}(\mathscr{G})$.

Then, $\mathscr{G}'' = \langle \Sigma_0', \Sigma_1, \mathscr{L} \circ \mathscr{L}', s' \rangle$ is a lexicalized ACG by the first condition and we see that $\mathcal{O}(\mathscr{G}'') = \mathcal{O}(\mathscr{G})$ by the second condition.

[2]The idea presented in this subsection is based on a discussion among speakers, Ph. de Groote, S. Salvati, R. Muskens, M. Kanazawa, and Yoshinaka, of the *First Workshop on Lambda Calculus and Formal Grammar* held in 2005, Tokyo.

For defining Σ_0' and $\mathscr{L}'$, in fact it is enough to find an integer $k_{\mathscr{G}} \in \mathbb{N}$ such that for every $M \in \mathcal{A}(\mathscr{G})$, there is $N \in \mathcal{A}(\mathscr{G})$ with $N \approx M$ and

$$\#_{\mathscr{C}_0^-}(N) \leq k_{\mathscr{G}} \cdot \#_{\mathscr{C}_0^+}(N). \tag{4.1}$$

Suppose that such an integer $k_{\mathscr{G}}$ exists. Define Σ_0' and $\mathscr{L}' : \Sigma_0' \to \Sigma_0$ by

$$\begin{aligned}
\mathscr{A}_0' &= \mathscr{A}_0, \\
\mathscr{C}_0' &= \{\, [\![\mathsf{a}, \mathsf{c}_1, \ldots, \mathsf{c}_k]\!] \mid 0 \leq k \leq k_{\mathscr{G}},\ \mathsf{a} \in \mathscr{C}_0^+,\ \mathsf{c}_i \in \mathscr{C}_0^- \,\} \\
\tau_0'([\![\mathsf{a}, \mathsf{c}_1, \ldots, \mathsf{c}_k]\!]) &= (\tau_0(\mathsf{a}) \to \tau_0(\mathsf{c}_1) \to \cdots \to \tau_0(\mathsf{c}_k) \to s) \to s \\
\mathscr{L}'(p) &= p \text{ for } p \in \mathscr{A}_0, \\
\mathscr{L}'([\![\mathsf{a}, \mathsf{c}_1, \ldots, \mathsf{c}_k]\!]) &= \lambda w.w\mathsf{a}\mathsf{c}_1 \ldots \mathsf{c}_k.
\end{aligned}$$

Then, for $\mathscr{G}' = \langle \Sigma_0', \Sigma_1, \mathscr{L} \circ \mathscr{L}', s \rangle$, we see that $\mathcal{O}(\mathscr{G}) = \mathcal{O}(\mathscr{G}')$. The inclusion $\mathcal{O}(\mathscr{G}') \subseteq \mathcal{O}(\mathscr{G})$ is trivial, since for every $P \in \mathcal{A}(\mathscr{G}')$ we have $|\mathscr{L}'(P)|_{\beta\eta} \in \mathcal{A}(\mathscr{G})$. We show the converse. For $M \in \mathcal{A}(\mathscr{G})$, by the hypothesis we can assume that

$$\#_{\mathscr{C}_0^-}(M) \leq k_{\mathscr{G}} \cdot \#_{\mathscr{C}_0^+}(M).$$

Let M' be obtained from M by replacing each occurrence of a constant by a fresh variable so that

$$M = M'[\mathsf{a}_1/x_1, \ldots, \mathsf{a}_m/x_m, \mathsf{c}_1/y_1, \ldots, \mathsf{c}_n/y_n]$$

where each a_i is lexical and c_i is nonlexical. Since $n \leq k_{\mathscr{G}} \cdot m$, we can partition the set $\{1, \ldots, n\}$ into m (possibly empty) subsets $I_1, \ldots, I_m$ such that $|I_i| \leq k_{\mathscr{G}}$. Let

$$P = [\![\mathsf{a}_1, \vec{\mathsf{c}}_{I_1}]\!](\lambda x_1 \vec{y}_{I_1}.[\![\mathsf{a}_2, \vec{\mathsf{c}}_{I_2}]\!](\lambda x_2 \vec{y}_{I_2}. \ldots [\![\mathsf{a}_m, \vec{\mathsf{c}}_{I_m}]\!](\lambda x_m \vec{y}_{I_m}.M') \ldots))$$

where $\vec{y}_{I_i}$ is a sequence of variables y_j with $j \in I_i$ and $\vec{\mathsf{c}}_{I_i}$ is the corresponding sequence of c_j with $j \in I_i$. Then we have $P \in \mathcal{A}(\mathscr{G}')$, $\mathscr{L}'(P) = M$ and thus $\mathcal{O}(\mathscr{G}') \subseteq \mathcal{O}(\mathscr{G})$. Note that the above construction increases the order of the abstract vocabulary by 2. Although this construction will have to be modified, explaining this basic idea should help the reader to understand our lexicalization method.

How can we find that key integer $k_{\mathscr{G}}$? In fact, if the given ACG $\mathscr{G}$ contains no nullary or unary nonlexical constants, then $k_{\mathscr{G}}$ is given as

$$k_{\mathscr{G}} = \max\{\, |\tau_0(\mathsf{a})| \mid \mathsf{a} \in \mathscr{C}_0^+ \,\}. \tag{4.2}$$

Our algorithm first transforms the given semilexicalized ACG into an equivalent one that satisfies the assumption that every nonlexical constant is neither

nullary nor unary, but, we postpone the discussion of this transformation until later. We here explain why the number given by (4.2) is enough under the assumption. First we introduce the notion of a *type occurrence in a term*, by explicitly displaying the type of every subterm occurring in a term, like we do with variables. For instance, if M has type $\gamma \to \delta$ and N has type γ, then we write $(M^{\gamma\to\delta} N^{\gamma})^{\delta}$ instead of MN. If an atomic type p occurs in γ, we have two corresponding occurrences of p in $(M^{\gamma\to\delta} N^{\gamma})^{\delta}$, one in the type $\gamma \to \delta$ of M and one in the type γ of N. If an atomic type q occurs in δ, we have two corresponding occurrences of q, one in the type $\gamma \to \delta$ of M and one in the type δ of MN. We then draw *links* connecting corresponding atomic type occurrences as

$$(M^{\gamma[p]\to\delta[q]} N^{\gamma[p]})^{\delta[q]}$$

where $\gamma[p]$ specifies an occurrence of the type p in γ and likewise for $\delta[q]$. For an abstraction term $(\lambda x^{\gamma}.M^{\delta})^{\gamma\to\delta}$, since M has exactly one occurrence of the free variable x^{γ} by the linearity, we can give links as follows

$$(\lambda x^{\gamma}.M^{\delta[q]}[x^{\gamma[p]}])^{\gamma[p]\to\delta[q]}$$

where $M[x]$ specifies the occurrence of the variable x in M. Though the above two examples illustrate links of length just one, links form longer paths in a term. Consequently, for instance, the occurrence of p in the type of c is linked to the one in the type of y in

$$y^{o\to(p\to q)\to r}\mathsf{a}^{o}(\lambda x^{p}.\mathsf{c}^{p\to q\to p}x^{p})^{p\to q}.$$

Here we suppress the types of some subterms for conciseness. Actually the path illustrated above passes through the occurrence of p in $(y\mathsf{a})^{(p\to q)\to r}$. Note that in a linear term M, no path branches and every path has exactly two end-points.

For conciseness, if the intended occurrence is clear, by "term" we mean an appropriate occurrence of that term, and "type" we mean an appropriate occurrence of that type.

Suppose that a β-normal term M is in $\mathcal{A}(\mathscr{G})$. To compare the numbers of occurrences of lexical and nonlexical constants, in other words, to make sure that $\#_{\mathscr{C}_0^-}(M) \leq k_{\mathscr{G}} \cdot \#_{\mathscr{C}_0^+}(M)$ for $k_{\mathscr{G}}$ defined as (4.2), we divide M into one lexical part M_0 and several nonlexical parts $M_1, \ldots, M_m$ so that

1. $M = (\lambda x_1 \ldots x_m.M_0)M_1 \ldots M_m$,
2. $M_0 \in \Lambda(\Sigma_0^+)$ and M_0 is in long normal form,
3. $M_i \in \Lambda(\Sigma_0^-)$ and M_i is in long normal form for $i \geq 1$,

4. for each $i \geq 1$, M_i has the form $M_i = \lambda\vec{z}_i.M_i'$, where $\vec{z}_i$ consists of variables of atomic types, and M_i' also has an atomic type and contains no variables other than $\vec{z}_i$,

5. m is as small as possible.

The last condition forbids M_0 to have a subterm of the form $x_i\vec{N}_1(x_j\vec{N}_2)$. If M_0 has such a subterm, we can decrease the number m of nonlexical parts by replacing the subterm with $x_{i,j}\vec{N}_1\vec{N}_2$ and combining two nonlexical parts M_i and M_j as follows:

$$M_{i,j} = \lambda\vec{z}_1\vec{z}_2.M_i\vec{z}_1(M_j\vec{z}_2),$$

where $|\vec{z}_1| = |\vec{N}_1|$ and $|\vec{z}_2| = |\vec{N}_2|$. Of course each M_i cannot be $\lambda z.z$. For instance, if

$$M \equiv \mathsf{b}\Big(\lambda y_1^s y_2^s.\mathsf{c}\big(\mathsf{ca}(\mathsf{b}(\lambda y_0^s.y_0))\big)\big(\mathsf{b}(\lambda y_3^s y_4^s.\mathsf{c}(\mathsf{c}y_1y_2)(\mathsf{c}y_3y_4))\big)\Big),$$

where $\mathsf{a}, \mathsf{b} \in \mathscr{C}_0^+$, $\mathsf{c} \in \mathscr{C}_0^-$, $\tau_0(\mathsf{a}) = s$, $\tau_0(\mathsf{b}) = (s \to s) \to s$, and $\tau_0(\mathsf{c}) = s^2 \to s$, we let

$$\begin{aligned}
M_0 &\equiv \mathsf{b}\Big(\lambda y_1^s y_2^s.x_1^{s^3\to s}\mathsf{a}\big(\mathsf{b}(\lambda y_0^s.y_0)\big)\big(\mathsf{b}(\lambda y_3^s y_4^s.x_2^{s^4\to s}y_1y_2y_3y_4)\big)\Big),\\
M_1 &\equiv \lambda z_{1,1}^s z_{1,2}^s z_{1,3}^s.\mathsf{c}(\mathsf{c}z_{1,1}z_{1,2})z_{1,3},\\
M_2 &\equiv \lambda z_{2,1}^s z_{2,2}^s z_{2,3}^s z_{2,4}^s.\mathsf{c}(\mathsf{c}z_{2,1}z_{2,2})(\mathsf{c}z_{2,3}z_{2,4}).
\end{aligned}$$

Suppose that a nonlexical part M_i with $i \geq 1$ has the form $M_i = \lambda z_1^{p_1}\dots z_n^{p_n}.M_i'$ with $\tau_0(M_i) = p_1 \to \dots \to p_n \to q$. M_i' can be regarded as a tree over $\mathscr{C}_0^- \cup \{z_1, \dots, z_n\}$. Since M contains no nullary or unary nonlexical constants, every constant in M_i' has at least two arguments. If a tree contains no node of rank 1, then the number of leaves is larger than the number of internal nodes. In M_i, every leaf node is a variable and every internal node is a nonlexical constant. Thus, we have

$$1 \leq \#_{\mathscr{C}_0^-}(M_i) < n = |\tau_0(M_i)| - 1. \tag{4.3}$$

On the other hand, the type of x_i in M_0 is $\tau_0(M_i) = p_1 \to \dots \to p_n \to q$. Now we are going to show that

$$\begin{cases} \textit{the path from each } p_j \textit{ in the type of } x_i \\ \textit{ends in the type of a lexical constant in } M_0. \end{cases} \tag{4.4}$$

Since M_0 is in long normal form, the occurrence of x_i in M_0 has exactly n arguments $N_1, \dots, N_n$. Since the type p_j of N_j is atomic, N_j is not an

abstraction term. By the fifth condition, the head of N_j cannot be one of the free variables $x_1, \dots, x_m$ of M_0. If the head of N_j is a lexical constant, (4.4) holds. Otherwise, it is a bound variable $w_0^{\vec{\alpha}_0 \to p_j}$, where we find a path

$$x_i^{p_1 \to \cdots \to \overbrace{p_j \to \cdots \to p_n \to q}} N_1 \dots N_{j-1} (w_0^{\vec{\alpha}_0 \to p_j} \vec{N}) N_{j+1} \dots N_n.$$

The path goes to the type of the abstraction term $L_0^{(\vec{\alpha}_0 \to p_j) \to \beta_0}$ that binds the variable w_0,

$$L_0 = \big(\lambda w_0^{\vec{\alpha}_0 \to p_j}.L[x_i^{p_1 \to \cdots \to \overbrace{p_j \to \cdots \to p_n \to q}} N_1 \dots N_{i-1} (w_0^{\vec{\alpha}_0 \to \overbrace{p_j}} \vec{N}) N_{i+1} \dots N_n / z]\big)^{(\vec{\alpha}_0 \to \overbrace{p_j}) \to \beta_0}$$

Since M_0 has the distinguished type s, L_0 of type $(\vec{\alpha}_0 \to p_j) \to \beta_0$ cannot be M_0. Since M_0 is β-normal, L_0 cannot have an argument. Thus we can find the smallest application term K_0 which has L_0 as a proper subterm. K_0 has the form

$$K_0 = \mathsf{x}^{\vec{\gamma} \to (\vec{\beta} \to (\vec{\alpha}_0 \to \overbrace{p_j}) \to \beta_0) \to \delta_0} \vec{L}_1^{\vec{\gamma}} (\lambda \vec{v}^{\vec{\beta}}.L_0^{(\vec{\alpha}_0 \to \overbrace{p_j}) \to \beta_0}).$$

If x is a lexical constant, (4.4) holds. Otherwise, since all free variables $x_1, \dots, x_m$ of M_0 have a second-order type, x must be a bound variable. Let $\mathsf{x} = w_1^{\vec{\gamma} \to (\vec{\beta} \to (\vec{\alpha}_0 \to p_j) \to \beta_0) \to \delta_0}$ and find the unique abstraction term that binds the occurrence of w_1. This way we can trace the path from the occurrence of p_j in the type of x_i. If the path passes through the types of variables $x_i, w_0, w_1, \dots$ in this order, then the type of w_{l+1} is more complicated than that of x_l, i.e., $\mathrm{ord}(\tau_0(w_{l+1})) \geq \mathrm{ord}(\tau_0(w_l)) + 2$. Since M_0 is finite, eventually we see that the path ends in a type occurrence of a lexical constant. We have proved the statement (4.4).

Suppose that the path from p_j in the type of x_i ends in the type $\alpha[p_j]$ of a lexical constant a. When conversely we start tracing the path from the occurrence p_j in $\alpha[p_j] = \tau_0(\mathsf{a})$, it cannot end in any other place than the occurrence p_j in the type $p_1 \to \cdots \to p_n \to q$ of x_i. Therefore,

$$\sum_{1 \leq i \leq m} \big(|\tau_0(x_i)| - 1\big) \leq \sum_{\mathsf{a} \in \mathscr{C}_0^+} \#_{\mathsf{a}}(M) |\tau_0(\mathsf{a})|.$$

By (4.3) and the definition (4.2) of the key integer $k_{\mathscr{G}}$, we have

$$\#_{\mathscr{C}_0^-}(M) = \sum_{1 \leq i \leq m} \#_{\mathscr{C}_0^-}(M_i) < \sum_{1 \leq i \leq m} \big(|\tau_0(M_i)| - 1\big) = \sum_{1 \leq i \leq m} \big(|\tau_0(x_i)| - 1\big)$$
$$\leq \sum_{\mathsf{a} \in \mathscr{C}_0^+} \#_{\mathsf{a}}(M) |\tau_0(\mathsf{a})| \leq \#_{\mathscr{C}_0^+}(M) \cdot \max\{\, |\tau_0(\mathsf{a})| \mid \mathsf{a} \in \mathscr{C}_0^+ \,\} = k_{\mathscr{G}} \cdot \#_{\mathscr{C}_0^+}(M).$$

Remark 4.11. We note some properties of a path here.

- In $\mathsf{x}M_1 \dots M_n$ where x is an atomic term, if a path starts from inside M_i, then it cannot go into any other M_j with $j \neq i$.
- If both two end-points of a path are inside the types of some constants, then one point is at a positive occurrence of the atomic type, and the other is at a negative occurrence of the atomic type.

4.3.2 First Step

Our actual lexicalization algorithm consists of three steps: firstly we convert $\mathscr{G}$ into an ACG satisfying the conditions in Definition 4.12, secondly we eliminate nullary and unary nonlexical constants, and finally all the nonlexical constants are eliminated. This section is devoted to the first step. The treatment of remaining steps varies depending on the order of the given semilexicalized ACG.

The conditions in Definition 4.12 talk about the nonlexical part of the abstract vocabulary.

Definition 4.12 (Condition I & II). A semilexicalized ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$ *satisfies Condition I* if for every second-order closed term $M \in \Lambda(\Sigma_0^-)$, there is $N \approx M$ such that

i. if $\tau_0(M)$ is nullary, then $N \in \mathscr{C}_0^-$,

ii. if $\tau_0(M)$ is not nullary, then N contains no nullary nonlexical constants.

A semilexicalized ACG $\mathscr{G}$ *satisfies Condition II*, if for every second-order closed term $M \in \Lambda(\Sigma_0^-)$, there is a β-normal term N such that $N \approx M$ and

i. if $\tau_0(M)$ is unary, then $N \in \mathscr{C}_0^-$,

ii. if N contains a unary constant $\mathsf{c} \in \mathscr{C}_0^-$, then the argument of c is a bound variable (if any).

Lemma 4.13. *Suppose that a semilexicalized ACG $\mathscr{G}$ satisfies Condition I. Then for every $M \in \Lambda(\Sigma_0)$, there is a β-normal term $N \in \Lambda(\Sigma_0)$ such that $N \approx M$ and if a nullary nonlexical constant c occurs in N, then either*

- $N = \mathsf{c}$,
- c *occurs as an argument of a lexical constant, or*
- c *occurs as an argument of a variable.*

Proof. Let us divide M into one lexical part M_0 and several nonlexical parts $M_1, \ldots, M_m$ so that

1. $M = (\lambda x_1 \ldots x_m.M_0)M_1 \ldots M_m$,
2. $M_0 \in \Lambda(\Sigma_0^+)$ and M_0 is in long normal form,
3. $M_i \in \Lambda(\Sigma_0^-)$ and M_i is in long normal form for $i \geq 1$,
4. for each $i \geq 1$, M_i has the form $M_i = \lambda \vec{z_i}.M_i'$, where $\vec{z_i}$ consists of variables of atomic types, and M_i' also has an atomic type and contains no variables other than $\vec{z_i}$,
5. m is as small as possible.

By Condition I, there are $N_1, \ldots, N_m$ such that $N_i \approx M_i$ and either N_i is a nullary nonlexical constant or N_i contains no nullary nonlexical constants. Let $N = |M_0[N_1/x_1, \ldots, N_m/x_m]|_\beta$ and suppose that N_j is a nullary nonlexical constant c. If $M_0 = x_j$, then $N = N_j = \mathsf{c}$. If $M_0 \neq x_j$, there is M_0' and an atomic term x such that $M_0 = M_0'[\mathsf{x}\vec{L}_1 x_j/z]$, since x_j has an atomic type. Since M_0 has no subterm of the form $x_i\vec{N}_1(x_j\vec{N}_2)$, x is either a lexical constant or a variable other than $x_1, \ldots, x_m$. In N, N_j appears as an argument of x. □

Similarly we have the following lemma.

Lemma 4.14. *Suppose that a semilexicalized ACG $\mathscr{G}$ satisfies Condition II. Then for every $M \in \Lambda(\Sigma_0)$, there is a β-normal term $N \in \Lambda(\Sigma_0)$ such that $N \approx M$ and if a unary nonlexical constant c occurs in a subterm $\mathsf{c}N'$ of N, then the head of N' is either a lexical constant or a variable.*

Note that Condition II-i ensures that for every atomic type p, there is a unary nonlexical constant $\mathsf{e}_p \in \mathscr{C}_0^-$ of type $p \to p$ which is mapped to the identity $\lambda x^{\mathscr{L}(p)}.x$.

First we present the procedure for converting a semilexicalized ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s\rangle$ into an equivalent ACG $\mathscr{G}' = \langle \Sigma_0', \Sigma_1, \mathscr{L}', s\rangle$ satisfying Condition I in Definition 4.12. We define the new abstract vocabulary Σ_0' and lexicon $\mathscr{L}'$ as extensions of Σ_0 and $\mathscr{L}$ by adding new nonlexical constants. This procedure is performed on the nonlexical part Σ_0^- of the abstract vocabulary and we ignore the lexical part Σ_0^+.

Let $\Gamma_{\mathscr{G}} \subseteq \mathscr{A}_0 \times \{\, P \in \Lambda(\Sigma_1) \mid P \text{ is a combinator} \,\}$ be defined as follows:

$$\begin{aligned}
\Gamma_0 &= \varnothing, \\
\Gamma_{n+1} &= \Gamma_n \cup \{\, \langle\, q,\, |\mathscr{L}(\mathsf{c})P_1 \dots P_m|_{\beta\eta} \,\rangle \mid \\
&\qquad\qquad \mathsf{c} \in \mathscr{C}_0^-,\ \tau_0(\mathsf{c}) = p_1 \to \cdots \to p_m \to q,\ \langle p_i, P_i\rangle \in \Gamma_n \,\}, \\
\Gamma_{\mathscr{G}} &= \bigcup_{n\geq 1} \Gamma_n.
\end{aligned}$$

Recall that for a fixed type, only finitely many linear combinators of that type exist (modulo $\beta\eta$). Thus, $\Gamma_{\mathscr{G}}$ is finite, because $\tau_1(P) = \mathscr{L}(p)$ for every element $\langle p, P\rangle \in \Gamma_{\mathscr{G}}$. Obviously each Γ_n can be computed effectively. If $\Gamma_n = \Gamma_{n+1}$, then $\Gamma_n = \Gamma_{n+k}$ for all $k \geq 0$ and thus $\Gamma_{\mathscr{G}} = \Gamma_n$. Therefore, $\Gamma_{\mathscr{G}}$ can be computed effectively. Moreover, immediately by the construction of $\Gamma_{\mathscr{G}}$, for every variable-free term $M \in \Lambda(\Sigma_0^-)$ of an atomic type, we have $\langle \tau_0(M), |\mathscr{L}(M)|_{\beta\eta}\rangle \in \Gamma_{\mathscr{G}}$.

To satisfy Condition I-i, for each $\langle p, P\rangle \in \Gamma_{\mathscr{G}}$, let us put the following lexical entries[3] into $\mathscr{C}_0'$

$$\langle [\![p, P]\!],\ p,\ P \rangle.$$

Next, for Condition I-ii, we add the following lexical entries

$$\langle [\![\vec{p} \to q,\ |\lambda\vec{x}.\mathscr{L}(\mathsf{c})P_1 \dots P_m|_{\beta\eta}]\!],\ \vec{p} \to q,\ \lambda\vec{x}.\mathscr{L}(\mathsf{c})P_1 \dots P_m \rangle$$

where

- $\mathsf{c} \in \mathscr{C}_0^-$,
- $\tau_0(\mathsf{c}) = p_1 \to \cdots \to p_m \to q$,
- either $\langle p_i, P_i\rangle \in \Gamma_{\mathscr{G}}$ or $P_i = x_i$,
- $\vec{p}$ and $\vec{x}$ are the subsequences of $p_1 \dots p_m$ and $x_1 \dots x_m$, respectively, such that p_i is in $\vec{p}$ iff x_i is in $\vec{x}$ iff $P_i = x_i$.

We obtain the following lemma.

Lemma 4.15 (Condition I). *Every semilexicalized ACG $\mathscr{G} \in \mathbf{G}(m,n)$ has an equivalent semilexicalized ACG $\mathscr{G}' \in \mathbf{G}(m,n)$ that satisfies Condition I in Definition 4.12.*

[3] Recall that a lexical entry of an ACG is a triple consisting of an abstract constant, its type, and the assigned object term.

Proof. Let $\mathscr{G}'$ be the ACG obtained from an ACG $\mathscr{G}$ by the above procedure. Since $\mathscr{G}'$ is an extension of $\mathscr{G}$, clearly $\mathcal{O}(\mathscr{G}) \subseteq \mathcal{O}(\mathscr{G}')$ holds. Moreover since the role of a new constant can be played by finitely many nonlexical constants of $\mathscr{G}$, the converse $\mathcal{O}(\mathscr{G}') \subseteq \mathcal{O}(\mathscr{G})$ holds. It is clear that for every $p \in \mathscr{A}_0$ and $P \in \Lambda(\Sigma_1)$ the following three are equivalent:

- there is a variable-free term $L \in \Lambda(\Sigma_0^-)$ of type p with $|\mathscr{L}(L)|_{\beta\eta} \equiv P$,
- $\langle p, P\rangle \in \Gamma_{\mathscr{G}}$,
- there is a nullary nonlexical constant of type p in $\mathscr{C}_0'^-$ that is mapped to P.

We show that $\mathscr{G}'$ satisfies Condition I. Since for every $M \in \Lambda(\Sigma_0'^-)$, there is $L \in \Lambda(\Sigma_0^-)$ such that $\tau_0(L) = \tau_0'(M)$ and $\mathscr{L}(L) = \mathscr{L}'(M)$, it is enough to show that for every closed second-order term $L \in \Lambda(\Sigma_0^-)$, there is $N \in \Lambda(\Sigma_0'^-)$ such that

- $\tau_0(L) = \tau_0'(N)$ and $\mathscr{L}(L) = \mathscr{L}'(N)$,

i. if $\tau_0(L)$ is nullary, then $N \in \mathscr{C}_0'^-$,

ii. if $\tau_0(L)$ is not nullary, then N contains no nullary nonlexical constants.

Condition I-i is satisfied by the above equivalence relation. We show that Condition I-ii is satisfied by induction on $\#_{\mathscr{C}_0^-}(L)$. Suppose that a closed second-order term $L \in \Lambda(\Sigma_0^-)$ has the form

$$L = \lambda\vec{y}.\mathsf{c}L_1 \dots L_m$$

with $\vec{y} \neq \varepsilon$ for a nonlexical constant c of type $p_1 \to \dots \to p_m \to q$. Let $\vec{y}_i$ be the subsequence of $\vec{y}$ whose elements appear in L_i and $I = \{\, i \mid \vec{y}_i \neq \varepsilon \,\}$. Note that $I \neq \varnothing$ by $\vec{y} \neq \varepsilon$. By the induction hypothesis and Condition I-i, for each $i \in \{1, \dots, m\}$ we have $N_i \in \Lambda(\Sigma_0'^-)$ such that

- $\tau_0'(N_i) = \tau_0(\lambda\vec{y}_i.L_i)$ and $\mathscr{L}'(N_i) = \mathscr{L}(\lambda\vec{y}_i.L_i)$
- $N_i \in \mathscr{C}_0'^-$ (and thus $\langle p_i, |\mathscr{L}'(N_i)|_{\beta\eta}\rangle \in \Gamma_{\mathscr{G}}$) if $i \notin I$,
- N_i contains no nullary constants if $i \in I$,

By the definition, we have the constant $[\![\vec{p}_I \to q, P]\!]$ where

- $P \equiv |\lambda\vec{x}_I.\mathscr{L}(\mathsf{c})P_1 \dots P_m|_{\beta\eta}$,
- $P_i = \mathscr{L}'(N_i)$ for $i \notin I$,

- $P_i = x_i$ for $i \in I$,
- $\vec{p}_I$ and $\vec{x}_I$ are the subsequences of $p_1 \dots p_m$ and $x_1 \dots x_m$ respectively such that p_i is in $\vec{p}_I$ iff x_i is in $\vec{x}_I$ iff $i \in I$.

Thus, for

$$N = \lambda \vec{y}.[\![\vec{p} \to q, P]\!]\langle N_i \vec{y}_i \rangle_{i \in I} \in \Lambda(\Sigma_0'^{-}),$$

we have $\tau_0'(N) = \tau_0(L)$ and $\mathscr{L}'(N) = \mathscr{L}(L)$. Since $I \neq \varnothing$, $[\![\vec{p}_I \to q, P]\!]$ is not nullary. N contains no nullary constants. □

The procedure for converting a semilexicalized ACG $\mathscr{G}$ into an equivalent ACG satisfying Condition II in Definition 4.12 is similar to the procedure for Condition I. We define an extension $\mathscr{G}'$ of $\mathscr{G}'$ assuming that the ACG $\mathscr{G}$ satisfies Condition I. Let $\Delta_{\mathscr{G}} \subseteq (\mathscr{A}_0 \to \mathscr{A}_0) \times \{ P \in \Lambda(\Sigma_1) \mid P \text{ is a combinator} \}$ be defined as follows:

$$\begin{aligned}
\Delta_0 &= \{ \langle p \to p, \lambda x^{\mathscr{L}(p)}.x \rangle \mid p \in \mathscr{A}_0 \} \\
\Delta_{n+1} &= \Delta_n \cup \{ \langle p \to r, |\lambda x.\mathscr{L}(\mathsf{c})(Px)|_{\beta\eta} \rangle \mid \mathsf{c} \in \mathscr{C}_0^{-}, \tau_0(\mathsf{c}) = q \to r, \\
&\qquad\qquad \langle p \to q, P \rangle \in \Delta_n \text{ for some } q \in \mathscr{A}_0 \} \\
\Delta_{\mathscr{G}} &= \bigcup_{n \geq 0} \Delta_n
\end{aligned}$$

By a similar reason to the reason for the computability of $\Gamma_{\mathscr{G}}$, $\Delta_{\mathscr{G}}$ can be computed effectively.

To satisfy Condition II-i, for each $\langle p \to q, P \rangle \in \Delta_{\mathscr{G}}$, we add the lexical entries

$$\langle [\![p \to q, P]\!],\ p \to q,\ P \rangle.$$

To satisfy Condition II-ii, furthermore we add the following constants

$$[\![p_1 \to \dots \to p_m \to r,\ |\lambda x_1 \dots x_m.P_0(\mathscr{L}(\mathsf{c})x_1 \dots x_m)|_{\beta\eta}]\!]$$

where

- $\mathsf{c} \in \mathscr{C}_0^{-}$,
- $\tau_0(\mathsf{c}) = p_1 \to \dots \to p_m \to q$,
- $\langle q \to r, P_0 \rangle \in \Delta_{\mathscr{G}}$.

Let $\tau_0'([\![\alpha, P]\!]) = \alpha$ and $\mathscr{L}'([\![\alpha, P]\!]) = P$. This way, we obtain the following lemma.

Lemma 4.16 (Condition II). *Every semilexicalized ACG $\mathscr{G} \in \mathbf{G}(m,n)$ has an equivalent semilexicalized ACG $\mathscr{G}' \in \mathbf{G}(m,n)$ that satisfies Conditions I and II in Definition 4.12.*

Proof. Let $\mathscr{G}'$ be the ACG obtained by the above procedure from an ACG $\mathscr{G}$, which satisfies Condition I. It is clear that $\mathcal{O}(\mathscr{G}) = \mathcal{O}(\mathscr{G}')$ and for every $p, q \in \mathscr{A}_0$ and $P \in \Lambda(\Sigma_1)$ the following three are equivalent:

- there is a closed term $M \in \Lambda(\Sigma_0^-)$ of type $p \to q$ with $\mathscr{L}(M) \equiv |P|_{\beta\eta}$,
- $\langle p \to q, P\rangle \in \Delta_{\mathscr{G}}$,
- there is a unary nonlexical constant of type $p \to q$ in $\mathscr{C}_0'^-$ that is mapped to P.

This equivalence relation entails that $\mathscr{G}'$ satisfies Condition II-i. To prove that $\mathscr{G}'$ satisfies Condition II, it is enough to show that for every closed term $L \in \Lambda(\Sigma_0^-)$ of type $\vec{p} \to r$ with $|\vec{p}| \geq 2$, there is $N \in \Lambda(\Sigma_0'^-)$ such that

- $\tau_0(L) = \tau_0'(N)$ and $\mathscr{L}(L) = \mathscr{L}'(N)$,
- if N contains a unary constant $\mathsf{c} \in \mathscr{C}_0^-$, then the argument of c is a bound variable.

We show that Condition II-ii is satisfied by induction on $\#_{\mathscr{C}_0^-}(L)$. Let a second-order closed term $L \in \Lambda(\Sigma_0^-)$ have the form $L = \lambda\vec{y}.\mathsf{d}\vec{L}$ where $|\vec{y}| \geq 2$ and $\tau_0(\mathsf{d}\vec{L}) \in \mathscr{A}_0$. Since $\mathscr{G}$ satisfies Condition I, we can assume that L contains no nullary constants. If $|\vec{L}| \geq 2$, the lemma holds immediately by the induction hypothesis. Suppose that $|\vec{L}| = 1$. L has the form

$$L = \vec{y}.\mathsf{d}_1(\dots(\mathsf{d}_n(\mathsf{c}L_1 \dots L_m))\dots)$$

where $\mathsf{d}_1 = \mathsf{d}$, all d_i are unary, and c is m-ary for some $m \geq 2$. By $|\vec{y}| \geq 2$, such a constant c occurs in L. Let $\tau_0(\mathsf{c}) = p_1 \to \dots \to p_m \to q$ and

$$P_0 = \mathscr{L}(\lambda z^q.\mathsf{d}_1(\dots(\mathsf{d}_n z)\dots)).$$

Since $\langle q \to r, P_0\rangle \in \Delta_{\mathscr{G}}$ for $q \to r = \tau_0(\lambda z.\mathsf{d}_1(\dots(\mathsf{d}_n z))$, $\mathscr{C}_0'$ has the constant

$$\mathsf{e} = [\![p_1 \to \dots \to p_m \to r,\ |\lambda x_1 \dots x_m.P_0(\mathscr{L}(\mathsf{c})x_1 \dots x_m)|_{\beta\eta}]\!].$$

Let $N_i \in \Lambda(\Sigma_0'^-)$ be obtained by applying the induction hypothesis to $\lambda\vec{y}_i.L_i$ where $\vec{y}_i$ is the subsequence of $\vec{y}$ whose elements appear in L_i. Then, the following N satisfies the condition:

$$N = \lambda\vec{y}.\mathsf{e}(N_1\vec{y}_1)\dots(N_m\vec{y}_m). \qquad \square$$

4.3.3 Second-Order Case

Though Yoshinaka and Kanazawa [61] have already stated that every second-order ACG has an equivalent lexicalized second-order ACG modulo combinators, the proof of that paper is in error. This subsection presents a correct lexicalization method for second-order ACGs. Another correction is shown by Kanazawa and Yoshinaka [27].

Elimination of Nullary and Unary Nonlexical Constants

Suppose that a given second-order ACG satisfies Conditions I and II of Definition 4.12. We eliminate nullary (*resp.* unary) nonlexical constants in a way similar to the procedure to make an second-order ACG satisfy Condition I-ii (*resp.* Condition II-ii). Let us add the following lexical entries to $\mathscr{G}$:

$$\langle [\![\lambda \vec{x}.\mathsf{a} P_1 \dots P_m]\!],\ \vec{p} \to q,\ \mathscr{L}(\lambda \vec{x}.\mathsf{a} P_1 \dots P_m)\rangle$$

where

- $\mathsf{a} \in \mathscr{C}_0^+$,
- $\tau_0(\mathsf{a}) = p_1 \to \dots \to p_m \to q$ and $\tau_0(P_i) = p_i$,
- either $P_i \in \mathscr{C}_0^-$ or $P_i = x_i$,
- p_i is in $\vec{p}$ iff x_i is in $\vec{x}$ iff $P_i = x_i$.

and

$$\langle [\![\lambda x_1 \dots x_m.\mathsf{c}(\mathsf{b} x_1 \dots x_m)]\!],\ p_1 \to \dots \to p_m \to r,\ \mathscr{L}(\lambda x_1 \dots x_m.\mathsf{c}(\mathsf{b} x_1 \dots x_m))\rangle$$

where

- $\mathsf{b} \in \mathscr{C}_0^+$,
- $\tau_0(\mathsf{b}) = p_1 \to \dots \to p_m \to q$,
- $\mathsf{c} \in \mathscr{C}_0^-$ of type $q \to r$.

This way, we obtain the following lemma.

Lemma 4.17. *Every ACG $\mathscr{G}$ belonging to $\mathbf{G}(2,n)$, has an equivalent (modulo combinators) ACG $\mathscr{G}' \in \mathbf{G}(2,n)$ which contains no nullary or unary nonlexical constants.*

Proof. Let $\mathscr{G}'$ be the ACG obtained from a second-order ACG $\mathscr{G}$ by the above procedure. It is enough to show by induction on $M \in \Lambda(\Sigma_0) - \Lambda(\Sigma_0^-)$ that if a variable-free term M has an atomic type, then there is $N \in \Lambda(\Sigma_0)$ such that $N \approx M$ and N contains no nullary or unary constants. The proof for Lemma 4.16 can be applied to this claim with few modification. Therefore, when we eliminate all the nullary and unary nonlexical constants from $\mathscr{G}'$, the object language does not shrink other than combinators. □

Elimination of Nonlexical Constants

If a second-order ACG $\mathscr{G} \in \mathbf{G}(2, n)$ has no nullary or unary nonlexical constants, then every $M \in \mathcal{A}(\mathscr{G})$ contains more occurrences of lexical constants than those of nonlexical constants. Therefore, as we have described in Section 4.3.1, the equation (4.1) holds for $k_{\mathscr{G}} = 1$. Let $\mathscr{G}' = \langle \Sigma_0', \Sigma_1, \mathscr{L} \circ \mathscr{L}', s \rangle \in \mathbf{G}^{\mathrm{lex}}(4, n)$ where

$\mathsf{x} \in \mathscr{C}_0'$	$\tau_0'(\mathsf{x})$	$\mathscr{L}'(\mathsf{x})$	for
a	$\tau_0(\mathsf{a})$	a	$\mathsf{a} \in \mathscr{C}_0^+$
$\mathsf{A_{ac}}$	$(\tau_0(\mathsf{c}) \to \tau_0(\mathsf{a}) \to s) \to s$	$\lambda x.x\mathsf{ca}$	$\mathsf{a} \in \mathscr{C}_0^+,\ \mathsf{c} \in \mathscr{C}_0^-$

However, this construction is not satisfactory because the order of $\mathscr{G}'$ is four. A naive alternative strategy would be to define τ_0' and $\mathscr{L}'$ so that

- For each variable-free term $M \in \Lambda(\Sigma_0)$ of an atomic type, there is $P \in \Lambda(\Sigma_0')$ such that $\mathscr{L}'(P) = M$.

This idea, however, does not work. Suppose that $M = \mathsf{c}M_1 \dots M_k$ for a nonlexical constant $\mathsf{c} \in \mathscr{C}_0^-$ and we have $P_1, \dots, P_k \in \Lambda(\Sigma_0')$ with $\mathscr{L}'(P_i) = M_i$ by the induction hypothesis. To construct M from $\mathscr{L}'(P_1), \dots, \mathscr{L}'(P_k)$, exactly one extra nonlexical constant c is required, but no lexical constant is needed. It is impossible if $\mathscr{G}'$ is lexicalized. We should use a lexical constant together whenever we use a nonlexical constant. Rather, consider the following statement:

- ⋆ *For each variable-free term $M \in \Lambda(\Sigma_0)$ of an atomic type, there are $P' \in \Lambda(\Sigma_0')$ and $\mathsf{a} \in \mathscr{C}_0^+$ such that $\tau_0(\mathsf{a}) \in \mathscr{A}_0$ and $\mathscr{L}'(P')\mathsf{a} = M$.*

Suppose that $M = \mathsf{c}M_1 \dots M_k$ for $\mathsf{c} \in \mathscr{C}_0^-$ and we have $\mathscr{L}'(P_i')\mathsf{a}_i = M_i$. Note that $k \geq 2$, since a nonlexical constant is neither nullary nor unary. Then,

$$\begin{aligned} M &= \mathsf{c}(\mathscr{L}'(P_1')\mathsf{a}_1) \dots (\mathscr{L}'(P_k')\mathsf{a}_k) \\ &= \big(\lambda x_1 \dots x_k z.\mathsf{c}x_1 \dots x_{k-2}(x_{k-1}\mathsf{a}_{k-1})(x_k z)\big) \\ &\qquad (\mathscr{L}'(P_1')\mathsf{a}) \dots (\mathscr{L}'(P_{k-2}')\mathsf{a}_{k-2})\mathscr{L}'(P_{k-1}')\mathscr{L}'(P_k')\mathsf{a}_k. \end{aligned}$$

If we have a constant $\mathsf{A} \in \mathscr{C}_0'$ such that

$$\mathscr{L}'(\mathsf{A}) = \lambda x_1 \dots x_k z.\mathsf{c} x_1 \dots x_{k-2}(x_{k-1}\mathsf{a}_{k-1})(x_k z),$$

then for $P' = \mathsf{A}_{\mathsf{ac}}(P_1'\mathsf{a}_1)\dots(P_{k-2}'\mathsf{a}_{k-2})P_{k-1}'P_k'$, we have $M = \mathscr{L}'(P')\mathsf{a}_{k-1}$. That is, we can use the statement $(\star)$ as an induction hypothesis.

Proposition 4.18. *Let $\mathscr{G}$ be a second-order ACG that contains no nullary or unary nonlexical constants. For the following third-order ACG $\mathscr{G}' = \langle \Sigma_0', \Sigma_0, \mathscr{L}', s\rangle$ with $\mathscr{L}'(p) = p$, we have $\mathcal{O}(\mathscr{G}') = \mathcal{A}(\mathscr{G})$:*

$\mathsf{x} \in \mathscr{C}_0'$	$\tau_0'(\mathsf{x})$	$\mathscr{L}'(\mathsf{x})$
a	$\tau_0(\mathsf{a})$	a
$\mathsf{D}_{\mathsf{ac}q'}$	$\vec{t} \to (p' \to p) \to (q' \to q) \to q \to r$	$\lambda \vec{w}^{\vec{t}} x^{p' \to p} y^{q' \to q} z^{q'}.\mathsf{c}\vec{w}(x\mathsf{a})(yz)$

where $\mathsf{a} \in \mathscr{C}_0^+$, $\mathsf{c} \in \mathscr{C}_0^-$, $\tau_0(\mathsf{a}) = p'$, $\tau_0(\mathsf{c}) = \vec{t} \to p \to q \to r$, and $q' \in \mathscr{A}_0$.

Proof. It is enough to prove the above statement $(\star)$.

Basis. If $M = \mathsf{a} \in \mathscr{C}_0^+$, let $P' = \lambda x^{\tau_0(\mathsf{a})}.x$.

Step. If $M = \mathsf{c}M_1 \dots M_k$ for $\mathsf{c} \in \mathscr{C}_0^-$, we can get P' that satisfies $(\star)$ as discussed above.

Suppose that $M = \mathsf{b}M_1 \dots M_k$ for $\mathsf{b} \in \mathscr{C}_0^+$ and $k \geq 1$. By the induction hypothesis, we have $P_1', \dots, P_k' \in \Lambda(\Sigma_0')$ and $\mathsf{a}_1, \dots, \mathsf{a}_k \in \mathscr{C}_0^+$ such that $M_i = \mathscr{L}'(P_i')\mathsf{a}_i$ for each $i \in \{1, \dots, k\}$. Let

$$P' = \lambda z.\mathsf{b}(P_1'\mathsf{a}_1)\dots(P_{k-1}'\mathsf{a}_{k-1})(P_k' z). \qquad \square$$

Corollary 4.19. *For every second-order ACG $\mathscr{G} \in \mathbf{G}(2, n)$, there is a third-order ACG $\mathscr{G}' \in \mathbf{G}^{\mathrm{lex}}(3, n)$ such that $\mathcal{O}(\mathscr{G}') = \{\, P \in \mathcal{O}(\mathscr{G}) \mid P$ contains a constant $\}$.*

This solution is not yet satisfactory, because the lexicalized ACG is third-order.

If every $P \in \Lambda(\Sigma_0')$ such that $\mathscr{L}'(P) = M$ for $M \in \mathcal{A}(\mathscr{G})$ can be constructed with application only, we can reduce the order of the abstract vocabulary to two. Since the usage of variables and λ-abstraction in the proof of Proposition 4.18 is tightly limit, we can modify $\mathscr{G}'$ into $\mathscr{G}''$ so that P' in the statement $(\star)$ can be constructed only by application. Let $\mathscr{G}'' = \langle \Sigma_0'', \Sigma_0, \mathscr{L}'', s\rangle$ consist of the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0''$	$\tau_0''(\mathsf{x})$	$\mathscr{L}''(\mathsf{x})$
e_p	$p \to p$	$\lambda x^p.x$
a	$\tau_0(\mathsf{a})$	a
$\mathsf{D}_{\mathsf{ac}q'}$	$\vec{t} \to (p' \to p) \to (q' \to q) \to q' \to r$	$\lambda \vec{w}^{\vec{t}} x^{p' \to p} y^{q' \to q} z^{q'}.\mathsf{c}\vec{w}(x\mathsf{a})(yz)$
$\mathsf{b}_{q'}$	$\vec{t} \to (q' \to q) \to q' \to r$	$\lambda \vec{w}^{\vec{t}} y^{q' \to q} z^{q'}.\mathsf{b}\vec{w}(yz)$

(4.5)

where $\mathsf{a} \in \mathscr{C}_0^+$, $\tau_0(\mathsf{a}) \in \mathscr{A}_0$, $\mathsf{c} \in \mathscr{C}_0^-$, $\tau_0(\mathsf{c}) = \vec{t} \to p \to q \to r$, $\mathsf{b} \in \mathscr{C}_0^+$, $\tau_0(\mathsf{b}) = \vec{t} \to q \to r$.
Then, replacing each subtype $(p \to q)$ with a new abstract atomic type $[p \to q]$[4], we get a second-order ACG. Although $\mathscr{L}''(\mathsf{e}_p)$ contains no lexical constants with respect to $\mathscr{L}$ and thus e_p is nonlexical with respect to $\mathscr{L} \circ \mathscr{L}''$, we can eliminate this nonlexical constant e_p in the same way as Lemma 4.17. This way, we obtain the following definition, where a further minor modification is added for the compactness of the new grammar.

Definition 4.20. Let a second-order ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle \in \mathbf{G}(2, n)$ contain no nullary or unary nonlexical constants. We define a second-order ACG $\mathscr{G}^l = \langle \Sigma_0', \Sigma_1, \mathscr{L} \circ \mathscr{L}', s \rangle \in \mathbf{G}(2, n+1)$ $(\mathscr{L}' : \Sigma_0' \to \Sigma_0)$ by

$$\mathscr{A}_0' = \mathscr{A}_0 \cup \{\, [p \to q] \mid p, q \in \mathscr{A}_0 \,\}$$
$$\mathscr{L}'(p) = p \text{ for } p \in \mathscr{A}_0, \text{ and } \mathscr{L}'([p \to q]) = p \to q \text{ for } p, q \in \mathscr{A}_0,$$

$\mathsf{x} \in \mathscr{C}_0'$	$\tau_0'(\mathsf{x})$	$\mathscr{L}'(\mathsf{x})$
$[\![\mathsf{a}]\!]$	p	a
$[\![\mathsf{a}_r]\!]$	$[p \to r] \to r$	$\lambda x^{p \to r}.x\mathsf{a}$

for $\mathsf{a} \in \mathscr{C}_0^+$, $\tau_0(\mathsf{a}) = p \in \mathscr{A}_0$, $r \in \mathscr{A}_0$,

$[\![\mathsf{b}]\!]$	$\vec{t} \to [q \to r]$	$\lambda \vec{w}^{\vec{t}} z^q.\mathsf{b}\vec{w}z$
$[\![\mathsf{b}_p]\!]$	$\vec{t} \to [p \to q] \to [p \to r]$	$\lambda \vec{w}^{\vec{t}} y^{p \to q} z^p.\mathsf{b}\vec{w}(yz)$

for $\mathsf{b} \in \mathscr{C}_0^+$, $\tau_0(\mathsf{b}) = \vec{t} \to q \to r$, $p \in \mathscr{A}_0$,

A_{ac}	$\vec{t} \to [q \to r]$	$\lambda \vec{w}^{\vec{t}} z^q.\mathsf{c}\vec{w}\mathsf{a}z$

for $\mathsf{a} \in \mathscr{C}_0^+$, $\tau_0(\mathsf{a}) = p$, $\mathsf{c} \in \mathscr{C}_0^-$, $\tau_0(\mathsf{c}) = \vec{t} \to p \to q \to r$,

$\mathsf{B}_{\mathsf{ac}q'}$	$\vec{t} \to [q' \to q] \to [q' \to r]$	$\lambda \vec{w}^{\vec{t}} y^{q' \to q} z^{q'}.\mathsf{c}\vec{w}\mathsf{a}(yz)$

for $\mathsf{a} \in \mathscr{C}_0^+$, $\tau_0(\mathsf{a}) = p$, $\mathsf{c} \in \mathscr{C}_0^-$, $\tau_0(\mathsf{c}) = \vec{t} \to p \to q \to r$, $q' \in \mathscr{A}_0$,

C_{ac}	$\vec{t} \to [p' \to p] \to [q \to r]$	$\lambda \vec{w}^{\vec{t}} x^{p' \to p} z^q.\mathsf{c}\vec{w}(x\mathsf{a})z$

for $\mathsf{a} \in \mathscr{C}_0^+$, $\tau_0(\mathsf{a}) = p'$, $\mathsf{c} \in \mathscr{C}_0^-$, $\tau_0(\mathsf{c}) = \vec{t} \to p \to q \to r$,

$\mathsf{D}_{\mathsf{ac}q'}$	$\vec{t} \to [p' \to p] \to [q' \to q] \to [q' \to r]$	$\lambda \vec{w}^{\vec{t}} x^{p' \to p} y^{q' \to q} z^{q'}.\mathsf{c}\vec{w}(x\mathsf{a})(yz)$

for $\mathsf{a} \in \mathscr{C}_0^+$, $\tau_0(\mathsf{a}) = p'$, $\mathsf{c} \in \mathscr{C}_0^-$, $\tau_0(\mathsf{c}) = \vec{t} \to p \to q \to r$, $q' \in \mathscr{A}_0$.

Lemma 4.21. $\mathcal{O}(\mathscr{G}^l) = \mathcal{O}(\mathscr{G})$.

[4]For instance, we replace $\mathsf{D}_{\mathsf{ac}q'}$ with two new constants that have different types $\vec{t} \to [p' \to p] \to [q' \to q] \to q' \to r$ and $\vec{t} \to [p' \to p] \to [q' \to q] \to [q' \to r]$.

Proof. The inclusion $\mathcal{O}(\mathscr{G}^l) \subseteq \mathcal{O}(\mathscr{G})$ is trivial. To see the converse $\mathcal{O}(\mathscr{G}) \subseteq \mathcal{O}(\mathscr{G}^l)$, we show that the following claim by induction on M.

For every variable-free term $M \in \Lambda(\Sigma_0)$ of an atomic type r, there is $P \in \Lambda(\Sigma_0')$ of type r such that $\mathscr{L}'(P) = M$. Moreover, if $M \notin \mathscr{C}_0$, there are $P' \in \Lambda(\Sigma_0')$ and $\mathsf{a} \in \mathscr{C}_0^+$ such that $\tau_0(\mathsf{a}) = p$, $\tau_0'(P') = [p \to r]$, and $\mathscr{L}'(P')\mathsf{a} = M$.

Basis. M is a constant. If $M = \mathsf{a} \in \mathscr{C}_0$, then $\mathsf{a} \in \mathscr{C}_0^+$, since $\mathscr{G}$ has no nullary nonlexical constants. Let $P = [\![\mathsf{a}]\!] \in \mathscr{C}_0'$.

Step. Case 1. The head of M is a lexical constant $\mathsf{b} \in \mathscr{C}_0^+$. Let $M = \mathsf{b}M_1 \dots M_k N$, $\tau_0(\mathsf{b}) = t_1 \to \dots \to t_k \to q \to r$, $\tau_0(M_i) = t_i$, and $\tau_0(N) = q$. By the induction hypothesis, there are $P_1, \dots, P_k \in \Lambda(\Sigma_0')$ such that $\tau_0'(P_i) = t_i$, and $\mathscr{L}'(P_i) = M_i$.

Case 1.1. $N = \mathsf{a} \in \mathscr{C}_0^+$. By the definition, there are constants $[\![\mathsf{b}]\!], [\![\mathsf{a}_r]\!] \in \mathscr{C}_0'$ such that

$$\begin{aligned} &\tau_0'([\![\mathsf{b}]\!]) = t_1 \to \dots \to t_k \to [q \to r], && \mathscr{L}'([\![\mathsf{b}]\!]) = \lambda w_1^{t_1} \dots w_k^{t_k} z^q.\mathsf{b}w_1 \dots w_k z, \\ &\tau_0'([\![\mathsf{a}_r]\!]) = [q \to r] \to r, && \mathscr{L}'([\![\mathsf{a}_r]\!]) = \lambda x^{q \to r}.x\mathsf{a}. \end{aligned}$$

Let $P' = [\![\mathsf{b}]\!]P_1 \dots P_k$ and $P = [\![\mathsf{a}_r]\!]P'$. We have $\tau_0(P') = [q \to r]$, $\tau_0(P) = r$, and

$$\mathscr{L}'(P) = \mathscr{L}'(P')\mathsf{a} = (\lambda z^q.\mathsf{b}M_1 \dots M_k z)\mathsf{a} = \mathsf{b}M_1 \dots M_k N = M.$$

Case 1.2. $N \notin \mathscr{C}_0$. By the induction hypothesis, there are $Q' \in \Lambda(\Sigma_0')$ and $\mathsf{a} \in \mathscr{C}_0^+$ such that $\tau_0'(Q') = [p \to q]$, $\tau_0(\mathsf{a}) = p$, and $\mathscr{L}'(Q')\mathsf{a} = N$. There is a constant $[\![\mathsf{b}_p]\!] \in \mathscr{C}_0'$ such that

$$\begin{aligned} \tau_0'(\mathsf{b}_p) &= t_1 \to \dots \to t_k \to [p \to q] \to [p \to r], \\ \mathscr{L}'(\mathsf{b}_p) &= \lambda w_1^{t_1} \dots w_k^{t_k} y^{p \to q} z^p.\mathsf{b}w_1 \dots w_k(yz) \end{aligned}$$

Let $P' = [\![\mathsf{b}_p]\!]P_1 \dots P_k Q'$ and $P = [\![\mathsf{a}_r]\!]P'$. We have $\tau_0'(P') = [p \to r]$, $\tau_0'(P) = r$, and

$$\mathscr{L}'(P) = \mathscr{L}'(P')\mathsf{a} = \mathsf{b}M_1 \dots M_k(\mathscr{L}'(Q')\mathsf{a}) = \mathsf{b}M_1 \dots M_k N = M.$$

Case 2. The head of M is a nonlexical constant $\mathsf{c} \in \mathscr{C}_0^-$. Let $M = \mathsf{c}M_1 \dots M_k N_1 N_2$ and $\tau_0(\mathsf{c}) = t_1 \to \dots \to t_k \to p \to q \to r$. By the induction hypothesis, there are $P_1, \dots, P_k$ such that $\tau_0'(P_i) = t_i$ and $\mathscr{L}'(P_i) = M_i$.

Case 2.1. $N_1 = \mathsf{a} \in \mathscr{C}_0^+$ and $N_2 = \mathsf{b} \in \mathscr{C}_0^+$. There is a constant $\mathsf{A}_{\mathsf{ac}} \in \mathscr{C}_0'$ such that

$$\begin{aligned} \tau_0'(\mathsf{A}_{\mathsf{ac}}) &= t_1 \to \dots \to t_k \to [q \to r], \\ \mathscr{L}'(\mathsf{A}_{\mathsf{ac}}) &= \lambda w_1^{t_1} \dots w_k^{t_k} z^q.\mathsf{c}w_1 \dots w_k\mathsf{a}z. \end{aligned}$$

Let $P' = \mathsf{A}_{\mathsf{ac}} P_1 \dots P_k$ and $P = [\![\mathsf{b}_r]\!] P'$. We have $\tau_0'(P') = [q \to r]$, $\tau_0'(P) = r$, and

$$\mathscr{L}'(P) = \mathscr{L}'(P')\mathsf{b} = \mathsf{c} M_1 \dots M_k \mathsf{a}\mathsf{b} = M.$$

Case 2.2. $N_1 = \mathsf{a} \in \mathscr{C}_0^+$ and $N_2 \notin \mathscr{C}_0^+$. By the induction hypothesis, we have Q_2' of type $[q' \to q]$ such that $\mathscr{L}'(Q_2')\mathsf{b} = N_2$ for some $\mathsf{b} \in \mathscr{C}_0^+$ of type q'. There is a constant $\mathsf{B}_{\mathsf{ac}q'} \in \mathscr{C}_0'$ such that

$$\tau_0'(\mathsf{B}_{\mathsf{ac}q'}) = t_1 \to \dots \to t_k \to [q' \to q] \to [q' \to r]$$
$$\mathscr{L}'(\mathsf{B}_{\mathsf{ac}q'}) = \lambda w_1^{t_1} \dots w_k^{t_k} y^{q' \to q} z^{q'}.\mathsf{c} w_1 \dots w_k \mathsf{a}(yz).$$

Let $P' = \mathsf{B}_{\mathsf{ac}q'} P_1 \dots P_k Q_2'$ and $P = [\![\mathsf{b}_r]\!] P'$. we have $\tau_0'(P') = [q' \to r]$, $\tau_0'(P) = r$ and

$$\mathscr{L}'(P) = \mathscr{L}'(P')\mathsf{b} = \mathsf{c} M_1 \dots M_k \mathsf{a}(\mathscr{L}'(Q_2')\mathsf{b}) = \mathsf{c} M_1 \dots M_k N_1 N_2 = M.$$

Case 2.3. $N_1 \notin \mathscr{C}_0^+$ and $N_2 = \mathsf{b} \in \mathscr{C}_0^+$. By the induction hypothesis, we have Q_1' of type $[p' \to p]$ such that $\mathscr{L}'(Q_1')\mathsf{a} = N_1$ for some $\mathsf{a} \in \mathscr{C}_0^+$ of type p'. There is a constant $\mathsf{C}_{\mathsf{ac}} \in \mathscr{C}_0'$ such that

$$\tau_0'(\mathsf{C}_{\mathsf{ac}}) = t_1 \to \dots \to t_k \to [p' \to p] \to [q \to r],$$
$$\mathscr{L}'(\mathsf{C}_{\mathsf{ac}}) = \lambda w_1^{t_1} \dots w_k^{t_k} x^{p' \to p} z^q.\mathsf{c} w_1 \dots w_k (x\mathsf{a}) z.$$

For $P' = \mathsf{C}_{\mathsf{ac}} P_1 \dots P_k Q_1'$ and $P = [\![\mathsf{b}_r]\!] P'$, we have $\tau_0'(P') = [q \to r]$, $\tau_0'(P) = r$ and

$$\mathscr{L}'(P) = \mathscr{L}'(P')\mathsf{b} = \mathsf{c} M_1 \dots M_k (\mathscr{L}'(Q_1')\mathsf{a})\mathsf{b} = \mathsf{c} M_1 \dots M_k N_1 N_2 = M.$$

Case 2.4. $N_1 \notin \mathscr{C}_0^+$ and $N_2 \notin \mathscr{C}_0^+$. By the induction hypothesis, we have $Q_1', Q_2' \in \Lambda(\Sigma_0')$ and $\mathsf{a}, \mathsf{b} \in \mathscr{C}_0^+$ such that

$$\tau_0'(Q_1') = [p' \to p], \qquad \mathscr{L}'(Q_1')\mathsf{a} = N_1, \qquad \tau_0(\mathsf{a}) = p',$$
$$\tau_0'(Q_2') = [q' \to q], \qquad \mathscr{L}'(Q_2')\mathsf{b} = N_2, \qquad \tau_0(\mathsf{b}) = q'.$$

By the definition of $\mathscr{G}^l$, there is a constant $\mathsf{D}_{\mathsf{ac}q'} \in \mathscr{C}_0'$ such that

$$\tau_0'(\mathsf{D}_{\mathsf{ac}q'}) = t_1 \to \dots \to t_k \to [p' \to p] \to [q' \to q] \to [q' \to r],$$
$$\mathscr{L}'(\mathsf{D}_{\mathsf{ac}q'}) = \lambda w_1^{t_1} \dots w_k^{t_k} x^{p' \to p} y^{q' \to q} z^{q'}.\mathsf{c} w_1 \dots w_k (x\mathsf{a})(yz).$$

Let $P' = \mathsf{D}_{\mathsf{ac}q'} P_1 \dots P_k Q_1' Q_2'$ and $P = [\![\mathsf{b}_r]\!] P'$. We have $\tau_0'(P') = [q' \to r]$, $\tau_0'(P) = r$ and

$$\mathscr{L}'(P) = \mathscr{L}'(P')\mathsf{b} = \mathsf{c} M_1 \dots M_k (\mathscr{L}'(Q_1')\mathsf{a})(\mathscr{L}'(Q_2')\mathsf{b}) = M. \qquad \square$$

Theorem 4.22. *For every second-order ACG $\mathscr{G} \in \mathbf{G}(2, n)$, there is a lexicalized second-order ACG $\mathscr{G}' \in \mathbf{G}(2, n+1)$ such that*

$$\mathcal{O}(\mathscr{G}') = \{\, R \in \mathcal{O}(\mathscr{G}) \mid R \text{ contains a constant} \,\}.$$

4.3.4 Fourth or Higher-Order Case

As we have mentioned at the beginning of Section 4.3.2, we start this section with elimination of nullary and unary nonlexical constants.

Elimination of Nullary and Unary Nonlexical Constants

Lemma 4.23 (Elimination of Nullary Nonlexical Constants). *For every semilexicalized ACG $\mathscr{G} \in \mathbf{G}(m,n)$, we can find a semilexicalized ACG $\mathscr{G}' \in \mathbf{G}(\max\{3,m\},n)$ such that $\mathscr{G}'$ contains no nullary nonlexical constants and*

$$\mathcal{O}(\mathscr{G}') = \{\, R \in \mathcal{O}(\mathscr{G}) \mid R \text{ contains a constant} \,\}.$$

Proof. By Lemma 4.16, we can assume that a given ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s\rangle$ satisfies Conditions I and II in Definition 4.12. We let an ACG $\mathscr{G}' = \langle \Sigma_0', \Sigma_1, \mathscr{L}', s\rangle$ have the following lexical entries, where $\mathscr{A}_0' = \mathscr{A}_0$ and $\mathscr{L}'(p) = \mathscr{L}(p)$ for $p \in \mathscr{A}_0$.

- $\langle \mathsf{c},\ \tau_0(\mathsf{c}),\ \mathscr{L}(\mathsf{c})\rangle$ for all $\mathsf{c} \in \mathscr{C}_0^-$ unless $\tau_0(\mathsf{c}) \in \mathscr{A}_0$,
- $\langle [\![M]\!],\ \tau_0(M),\ \mathscr{L}(M)\rangle$ for each M of the form

  $$M = \lambda x_1 \dots x_n.\mathsf{a}(x_1\mathsf{c}_{1,1}\dots\mathsf{c}_{1,m_1})\dots(x_n\mathsf{c}_{n,1}\dots\mathsf{c}_{n,m_n})$$

  where $\mathsf{a} \in \mathscr{C}_0^+$, $\tau_0(\mathsf{a}) = \gamma_1 \to \dots \to \gamma_n \to q$, $\mathsf{c}_{i,j} \in \mathscr{C}_0^-$, $\tau_0(\mathsf{c}_{i,j}) = p_{i,j}$, $\tau_0'(x_i) = p_{i,1} \to \dots \to p_{i,m_i} \to \gamma_i$ and $0 \le m_i \le |\gamma_i|$.

Since the order of the type of the bound variable x_i in the above M is

$$\operatorname{ord}(p_{i,1} \to \dots \to p_{i,m_i} \to \gamma_i) = \begin{cases} \max\{2, \operatorname{ord}(\gamma_i)\} & \text{if } m_i \ge 1 \\ \operatorname{ord}(\gamma_i) & \text{if } m_i = 0, \end{cases}$$

we have $\operatorname{ord}(\tau_0'([\![M]\!])) \le \max\{3, \operatorname{ord}(\tau_0(\mathsf{a}))\}$. Thus, $\mathscr{G}' \in \mathbf{G}(\max\{3,m\},n)$ if $\mathscr{G} \in \mathbf{G}(m,n)$. Moreover, $\mathscr{G}'$ also satisfies Condition II.

[$\mathcal{O}(\mathscr{G}') \subseteq \{\, R \in \mathcal{O}(\mathscr{G}) \mid R \text{ contains a constant} \,\}$]

The inclusion $\mathcal{O}(\mathscr{G}') \subseteq \mathcal{O}(\mathscr{G})$ is obvious. We show that $\mathcal{O}(\mathscr{G}')$ contains no combinators. Since every nonlexical constant $\mathsf{c} \in \mathscr{C}_0'^-$ has non-nullary second-order type, if $P \in \Lambda(\Sigma_0'^-)$ is closed, it cannot have an atomic type, in particular, the distinguished type. Thus if $P \in \mathcal{A}(\mathscr{G}')$, it contains a lexical constant.

[$\{\, R \in \mathcal{O}(\mathscr{G}) \mid R \text{ contains a constant} \,\} \subseteq \mathcal{O}(\mathscr{G}')$]

We say that an occurrence of a nullary nonlexical constant is *linked to* an occurrence of a lexical constant if the path which starts from the type of the

occurrence of the nullary nonlexical constant ends in a subtype of the type of the occurrence of the lexical constant. A term $M \in \Lambda(\Sigma_0)$ is said to satisfy the *linking condition* if for every occurrence of a nullary nonlexical constant, there is an occurrence of a lexical constant to which the occurrence of the nullary nonlexical constant is linked.

In fact, every $M \in \mathcal{A}(\mathscr{G}) - \Lambda(\Sigma_0^-)$ has an equivalent term $M' \approx M$ that satisfies the linking condition. Since $\mathscr{G}$ satisfies Condition I in Definition 4.12, we can assume that if a nullary constant c occurs in M, the occurrence is as an argument of a lexical constant or a bound variable, by Lemma 4.13. If the functor is a lexical constant a, then c is linked to the lexical constant a. If the functor is a bound variable, then as we have described in Section 4.3.1, we can trace the path until it ends in a subtype of the type of a lexical constant. Thus, here we assume that M satisfies the linking condition.

By induction on the size of M, we show that for every long normal term $M \in \Lambda(\Sigma_0)$ that satisfies the linking condition, there is $P \in \Lambda(\Sigma_0')$ with $\tau_0'(P) = \tau_0(M)$ and $\mathscr{L}'(P) = \mathscr{L}(M)$.

Case 1. Suppose that M is of the form $M = \lambda x.M'$. Since M satisfies the linking condition, also M' satisfies the linking condition. By the induction hypothesis, we have $P' \in \Lambda(\Sigma_0')$ with $\tau_0'(P') = \tau_0(M')$ and $\mathscr{L}'(P') = \mathscr{L}(M')$. For $P = \lambda x.P' \in \Lambda(\Sigma_0')$, we have $\tau_0'(P) = \tau_0(M)$ and $\mathscr{L}'(P) = \mathscr{L}(M)$.

Case 2. If M is of the form $M = \mathsf{c}$ for a nullary nonlexical constant c, then M cannot satisfy the linking condition.

Case 3. If M is of the form $M = \mathsf{x}M_1 \dots M_n$ for some variable or non-nullary nonlexical constant x, then each M_i satisfies the linking condition (see Remark 4.11). By the induction hypothesis, there is $P_i \in \Lambda(\Sigma_0')$ with $\tau_0'(P_i) = \tau_0(M_i)$ and $\mathscr{L}'(P_i) = \mathscr{L}(M_i)$. For $P = \mathsf{x}P_1 \dots P_n$, we have $\tau_0'(P) = \tau_0(M)$ and $\mathscr{L}'(P) = \mathscr{L}(M)$.

Case 4. The remaining case is that M is of the form

$$M = \mathsf{a}M_1 \dots M_n$$

for some lexical constant $\mathsf{a} \in \mathscr{C}_0^+$ of type $\gamma_1 \to \dots \to \gamma_n \to q$. For each $i \in \{1, \dots, n\}$, let M_i' be obtained from M_i by replacing with fresh variables $\vec{z}_i$ all the occurrences of nullary nonlexical constants linked to the head occurrence of a, so that M_i' satisfies the linking condition. By the induction hypothesis, there is P_i' such that $\tau_0'(P_i') = \tau_0(M_i')$ and $\mathscr{L}'(P_i') = \mathscr{L}(M_i')$. Let $\vec{\mathsf{c}}_i$ be such that

$$M_i = M_i'[\vec{\mathsf{c}}_i/\vec{z}_i].$$

By the property of paths, $|\vec{\mathsf{c}}_i| \leq |\gamma_i|$. By the definition of $\mathscr{G}'$, we have $[\![N]\!] \in \mathscr{C}_0'$ for

$$N = \lambda x_1 \dots x_n.\mathsf{a}(x_1\vec{\mathsf{c}}_1) \dots (x_n\vec{\mathsf{c}}_n).$$

Thus, for $P = [\![N]\!](\lambda\vec{z}_1.P'_1)\dots(\lambda\vec{z}_n.P'_n)$, we have $\tau'_0(P) = \tau_0(M)$ and

$$\mathscr{L}'(P) = \mathscr{L}\big(\,\mathsf{a}(M'_1[\vec{\mathsf{c}}_1/\vec{z}_1])\dots(M'_m[\vec{\mathsf{c}}_n/\vec{z}_n])\,\big) = \mathscr{L}(M). \qquad \square$$

Lemma 4.24 (Elimination of Unary Nonlexical Constants). *For every semilexicalized ACG $\mathscr{G} \in \mathbf{G}(m,n)$, we can find a semilexicalized ACG $\mathscr{G}' \in \mathbf{G}(\max\{4,m\},n)$ such that $\mathscr{G}'$ contains no nullary or unary nonlexical constants and*

$$\mathcal{O}(\mathscr{G}') = \{\, R \in \mathcal{O}(\mathscr{G}) \mid R \textit{ contains a constant} \,\}.$$

Proof. By Lemmas 4.16 and 4.23, we assume that $\mathscr{G}$ contains no nullary nonlexical constants and satisfies Condition II in Definition 4.12. $\mathcal{O}(\mathscr{G})$ contains no combinators.

We let an ACG $\mathscr{G}'$ have the following lexical entries:

- $\langle \mathsf{c},\ \tau_0(\mathsf{c}),\ \mathscr{L}(\mathsf{c})\rangle$ for all $\mathsf{c} \in \mathscr{C}_0^-$ unless $\tau_0(\mathsf{c})$ is unary,
- $\langle [\![M]\!],\ \tau_0(M),\ \mathscr{L}(M)\rangle$ for each M of the form

  $$M = \lambda x_1^{\alpha_{1,1}\to\dots\to\alpha_{1,m_1}\to\gamma_1}\dots x_n^{\alpha_{n,1}\to\dots\to\alpha_{n,m_n}\to\gamma_n}.\mathsf{c}_0(\mathsf{a}(x_1\vec{\mathsf{c}}_1)\dots(x_n\vec{\mathsf{c}}_n))$$

  where $\mathsf{a} \in \mathscr{C}_0^+$, $\tau_0(\mathsf{a}) = \gamma_1 \to \dots \to \gamma_n \to q$, $\vec{\mathsf{c}}_i = \mathsf{c}_{i,1}\dots\mathsf{c}_{i,m_i}$, $m_i \le |\gamma_i|$, and each $\mathsf{c}_{i,j}$ is a nonlexical constant of unary type $\alpha_{i,j}$.

Since the order of the type of the bound variables x_i in M is

$$\begin{aligned}&\operatorname{ord}(\alpha_{i,1}\to\dots\to\alpha_{i,m_i}\to\gamma_i)\\ =\ &\max\{\,\operatorname{ord}(\alpha_{i,j})+1,\ \operatorname{ord}(\gamma_i) \mid 1\le j\le m_i\,\} \le \max\{3,\operatorname{ord}(\gamma_i)\},\end{aligned}$$

we have $\operatorname{ord}(\tau'_0([\![M]\!])) \le \max\{4,\operatorname{ord}(\tau_0(\mathsf{a}))\}$. Thus, $\mathscr{G}' \in \mathbf{G}(\max\{4,m\},n)$ if $\mathscr{G} \in \mathbf{G}(m,n)$.

The inclusion $\mathcal{O}(\mathscr{G}') \subseteq \mathcal{O}(\mathscr{G})$ is trivial. We show the converse relation $\mathcal{O}(\mathscr{G}) \subseteq \mathcal{O}(\mathscr{G}')$.

We say that an occurrence of a unary nonlexical constant of type $p \to q$ is *linked to* an occurrence of a lexical constant if the path which starts from the negative occurrence of the subtype p in the type $p \to q$ of the occurrence of the unary nonlexical constant ends in p in the type of the occurrence of the lexical constant. A term $M \in \Lambda(\Sigma_0)$ is said to satisfy the *linking condition* if for every occurrence of a unary nonlexical constant, there is an occurrence of a lexical constant to which the occurrence of the unary nonlexical constant is linked.

In fact, every $M \in \mathcal{A}(\mathscr{G}) - \Lambda(\Sigma_0^-)$ has an equivalent term $M' \approx M$ that satisfies the linking condition. Since $\mathscr{G}$ satisfies Condition II in Definition 4.12, we can assume that if a unary constant c occurs in M, the head of the argument of c is either a lexical constant or a bound variable by Lemma 4.14. If it is a lexical constant a, then c is linked to the lexical constant a. If it is a bound variable, then as we have described in Section 4.3.1, we can trace the path until it ends in a subtype of the type of a lexical constant. Thus, here we assume that M satisfies the linking condition.

By induction on the size of M, we show that for every long normal term $M \in \Lambda(\Sigma_0)$ that satisfies the linking condition, there is $P \in \Lambda(\Sigma_0')$ with $\tau_0'(P) = \tau_0(M)$ and $\mathscr{L}'(P) = \mathscr{L}(M)$.

Case 1. $M = \lambda x.M'$ for some x. By the induction hypothesis we have $P' \in \Lambda(\Sigma_0')$ such that $\tau_0'(P') = \tau_0(M')$ and $\mathscr{L}'(P') = \mathscr{L}(M')$. Thus, for $P = \lambda x.P' \in \Lambda(\Sigma_0')$, we have $\tau_0'(P) = \tau_0(M)$ and $\mathscr{L}'(P) = \mathscr{L}(M)$.

Case 2. $M = \mathsf{x}M_1 \dots M_n$ where x is a variable or a non-unary nonlexical constant. By the induction hypothesis we have $P_i \in \Lambda(\Sigma_0')$ such that $\tau_0'(P_i) = \tau_0(M_i)$ and $\mathscr{L}'(P_i) = \mathscr{L}(M_i)$ for $1 \leq i \leq n$. Thus, for $P = \mathsf{x}P_1 \dots P_n \in \Lambda(\Sigma_0')$, we have $\tau_0'(P) = \tau_0(M)$ and $\mathscr{L}'(P) = \mathscr{L}(M)$.

Case 3. Suppose that the head of M is a unary nonlexical constant $\mathsf{c}_0 \in \mathscr{C}_0^-$ of type $p \to q$. Since M satisfies the linking condition, M is of the form

$$M = \mathsf{c}_0(\mathsf{a}M_1 \dots M_n)$$

for some lexical constant a of type $\gamma_1 \to \dots \to \gamma_n \to p$. Let M_i' be obtained from M_i by replacing with fresh variables $\vec{z}_i$ all the occurrences of unary nonlexical constants linked to the head occurrence of a, so that M_i' satisfies the linking condition. By the induction hypothesis, there is $P_i' \in \Lambda(\Sigma_0')$ such that $\tau_0'(P_i') = \tau_0(M_i')$ and $\mathscr{L}'(P_i') = \mathscr{L}(M_i')$. Let $\vec{\mathsf{c}}_i$ be such that

$$M_i = M_i'[\vec{\mathsf{c}}_i/\vec{z}_i].$$

By the property of paths, $|\vec{\mathsf{c}}_i| \leq |\gamma_i|$. By the definition of $\mathscr{G}'$, we have $[\![N]\!] \in \mathscr{C}_0'$ where

$$N = \lambda w_1 \dots w_n.\mathsf{c}_0(\mathsf{a}(w_1\vec{\mathsf{c}}_1) \dots (w_n\vec{\mathsf{c}}_n)).$$

Thus, for $P = [\![N]\!](\lambda\vec{z}_1.P_1') \dots (\lambda\vec{z}_n.P_n')$, we have

$$\begin{aligned}\mathscr{L}'(P) &= \mathscr{L}\big(\,\mathsf{c}_0(\mathsf{a}(M_1'[\vec{\mathsf{c}}_1/\vec{z}_1]) \dots (M_n'[\vec{\mathsf{c}}_n/\vec{z}_n]))\,\big)\\ &= \mathscr{L}(\mathsf{c}_0(\mathsf{a}M_1 \dots M_n)) = \mathscr{L}(M).\end{aligned}$$

Case 4. Suppose that the head of M is a lexical constant a of type $\gamma_1 \to \dots \to \gamma_n \to p$. M has form $M = \mathsf{a}M_1 \dots M_n$. Since $\mathscr{G}$ satisfies

Condition II-i, there is a nonlexical constant $\mathsf{e}_p \in \mathscr{C}_0$ of type $p \to p$ with $\mathscr{L}(\mathsf{e}_p) = \lambda x.x$, so $M \approx \mathsf{e}_p M$. This case is reduced to Case 3, where we use the induction hypothesis with respect to each M_i but not to $\mathsf{a}M_1 \dots M_n$. □

Elimination of Nonlexical Constants

Lemma 4.24 implies that every semilexicalized ACG has an equivalent lexicalized ACG as we have discussed in Section 4.3.1. The lexicalization method in Section 4.3.1 constructs a new ACG whose object vocabulary is the original abstract vocabulary and whose lexical entries have the form

$$\langle [\![\mathsf{a}, \vec{\mathsf{c}}]\!], (\tau_0(\mathsf{a}) \to \tau_0(\vec{\mathsf{c}}) \to s) \to s, \lambda w.w\mathsf{a}\vec{\mathsf{c}} \rangle,$$

where the bound variable w "grasps" and "controls" original abstract constants and then puts them at any places in a term we want. That trick is powerful, but causes the increase of the order of the grammar by two. To avoid this, we refrain from controlling lexical abstract constants, but continue to grasp nonlexical ones, by a bound variable in new lexical entries. Since the orders of nonlexical constants are at most two, this new strategy does not increase the order of the grammar if the order of the given semilexicalized ACG is four or higher.

Definition 4.25. Let a semilexicalized ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle \in \mathbf{G}(m, n)$ contain no nullary or unary nonlexical constants. Let $\Gamma \subseteq \Lambda(\Sigma_0)$ be the set of closed terms of the form

$$\lambda w_1 \dots w_n.\mathsf{a}(w_1\vec{\mathsf{c}}_1) \dots (w_n\vec{\mathsf{c}}_n) \quad \text{or}$$
$$\lambda w_0 w_1 \dots w_n.w_0\mathsf{c}_0(\mathsf{a}(w_1\vec{\mathsf{c}}_1) \dots (w_n\vec{\mathsf{c}}_n)))$$

where $\mathsf{a} \in \mathscr{C}_0^+$, c_0 and constants in $\vec{\mathsf{c}}_i$ are all nonlexical, $\tau_0(\mathsf{a}) = \gamma_1 \to \dots \to \gamma_n \to p$, $\tau_0(w_0) = \tau_0(\mathsf{c}_0) \to p \to q$ for some $q \in \mathscr{A}_0$, $\tau_0(w_i) = \tau_0(\vec{\mathsf{c}}_i) \to \gamma_i$ for $i \geq 1$, and $0 \leq |\vec{\mathsf{c}}_i| \leq |\tau_0(\gamma_i)|$. Note that Γ is a finite set.

The *lexicalized form of* $\mathscr{G}$ is an ACG $\mathscr{G}^l \in \mathbf{G}(\max\{4, m\}, n)$ that has the form

$$\mathscr{G}^l = \langle \Sigma_0', \Sigma_1, \mathscr{L} \circ \mathscr{L}', s \rangle$$

where

$$\begin{aligned} \mathscr{A}_0' &= \mathscr{A}_0, \\ \mathscr{C}_0' &= \{\, [\![M]\!] \mid M \in \Gamma \,\}, \\ \tau_0'([\![M]\!]) &= \tau_0(M), \\ \mathscr{L}'(p) &= p \text{ for } p \in \mathscr{A}_0, \\ \mathscr{L}'([\![M]\!]) &= M \text{ for } [\![M]\!] \in \mathscr{C}_0'. \end{aligned}$$

Example 4.26. Let $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$ have a lexical constant $\mathsf{a} \in \mathscr{C}_0^+$ of type $(s \to s) \to s$ and a nonlexical constant $\mathsf{c} \in \mathscr{C}_0^-$ of type $s^2 \to s$. Let us consider the terms M and $\mathsf{c}MM$ in $\mathcal{A}(\mathscr{G})$, where

$$M = \mathsf{a}(\lambda x_1^s.\mathsf{a}(\lambda x_2^s.\mathsf{a}(\lambda x_3^s.\mathsf{a}(\lambda x_4^s.\mathsf{c}(\mathsf{c}x_1x_2)(\mathsf{c}x_3x_4))))),$$

Let $\mathscr{G}' = \langle \Sigma_0', \Sigma_0, \mathscr{L}', s \rangle$ consist of the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0'$	$\tau_0'(\mathsf{x})$	$\mathscr{L}'(\mathsf{x})$
a	$(s \to s) \to s$	a
A	$\gamma \to s$	$\lambda w^\gamma.\mathsf{a}(w\mathsf{c})$
B	$\gamma \to \gamma \to s$	$\lambda w_0^\gamma w^\gamma.w_0\mathsf{c}\mathsf{a}(w\mathsf{c})$

where $\gamma = (s^2 \to s) \to s \to s$. For

$$\begin{aligned} N &= \mathsf{A}(\lambda y_1^{s^2 \to s} x_1^s.\mathsf{A}(\lambda y_2^{s^2 \to s} x_2^s.\mathsf{A}(\lambda y_3^{s^2 \to s} x_3^s.\mathsf{a}(\lambda x_4^s.y_1(y_2x_1x_2)(y_3x_3x_4))))), \\ N' &= \mathsf{B}(\lambda y_0^{s^2 \to s} z^s.y_0 z N) \\ &\qquad (\lambda y_1^{s^2 \to s} x_1^s.\mathsf{A}(\lambda y_2^{s^2 \to s} x_2^s.\mathsf{A}(\lambda y_3^{s^2 \to s} x_3^s.\mathsf{a}(\lambda x_4^s.y_1(y_2x_1x_2)(y_3x_3x_4))))), \end{aligned}$$

we have $\mathscr{L}'(N) = M$ and $\mathscr{L}'(N') = \mathsf{c}MM$.

Definition 4.27. Let $\mathscr{G}$ be a semilexicalized ACG that has no nullary or unary nonlexical constants. A β-expansion $(\lambda x_1 \dots x_m.M_0)M_1 \dots M_m$ of a β-normal term $M \in \Lambda(\Sigma_0)$ is *standard* if

- $M_0 \in \Lambda(\Sigma_0^+)$,
- $M_i \in \Lambda(\Sigma_0^-)$ for $i \in \{1, \dots, m\}$,
- every bound variable of M_i has an atomic type for $i \in \{1, \dots, m\}$,
- for every free variable y of M_i for $i \geq 1$, there is a nonlexical constant c with $\tau_0(y) = \tau_0(\mathsf{c})$.
- there is no subterm of the form $x_i\vec{N}_1(x_j\vec{N}_2)\vec{N}_3$ of M_0 for $i, j \in \{1, \dots, m\}$.

Let $\#(M_0, x_i)$ denote the number of negative occurrences of atomic types in the type $\tau_0(x_i)$ of the occurrence of x_i which are linked to a subtype of the type of some occurrence of a lexical constant in M_0. A standard β-expansion is *good* iff $\#_{\mathscr{C}_0^-}(M_i) \leq \#(M_0, x_i)$ holds for every M_i.

Note that every M_i (x_i) has a second-order type for $i \geq 1$.

Lemma 4.28. *For every $M \in \mathcal{A}(\mathscr{G})$, M has a good standard β-expansion.*

Proof. First we show that M has a standard β-expansion. Suppose that $M \in \mathcal{A}(\mathscr{G})$ has m occurrences of nonlexical constants $\mathsf{c}_1, \ldots, \mathsf{c}_m$. Let $M_0 \in \Lambda(\Sigma_0^+)$ be such that $M = (\lambda x_1 \ldots x_m.M_0)\mathsf{c}_1 \ldots \mathsf{c}_m$ and $M_i = \mathsf{c}_i$ for $i \in \{1, \ldots, m\}$. If $(\lambda x_1 \ldots x_m.M_0)M_1 \ldots M_m$ is a good β-expansion of M, the proof is done. Otherwise, M_0 has the form $M_0 = N_0[x_i\vec{N}_1(x_j\vec{N}_2)/w]$ for some $i, j \in \{1, \ldots, m\}$. Assume that $i = m-1$ and $j = m$. Let $M_0' = N_0[x_i\vec{N}_1\vec{N}_2/w]$ and $M_i' = \lambda\vec{z}_1\vec{z}_2.M_i\vec{z}_1(M_j\vec{z}_2)$, where $|\vec{z}_1| = |\vec{N}_1|$ and $|\vec{z}_2| = |\vec{N}_2|$. If $(\lambda x_1 \ldots x_{m-1}.M_0')M_1 \ldots M_{m-2}M_{m-1}'$ is a good β-expansion of M, the proof is done. Otherwise, by repeating this procedure, eventually we get a good β-expansion of M.

Let $(\lambda\vec{x}.M_0)\vec{M}$ be a standard β-expansion of M and $\tau_0(x_i) = p_{i,1} \to \cdots \to p_{i,n_i} \to p_i$. The occurrence of $p_{i,j}$ must be linked to some subtype of the type of an occurrence of a lexical constant, as we have described in Section 4.3.1. Thus $\#(M_0, x_i) = n_i$. On the other hand, the long normal form of M_i has the form $\lambda z_{i,1} \ldots z_{i,n_1}.M_i'$, where M_i' has an atomic type. M_i' can be regarded as a tree on $\mathscr{C}_0^- \cup \{z_{i,1}, \ldots, z_{i,n_i}\}$ whose leaves are exactly $z_{i,1}, \ldots, z_{i,n_i}$. Recall that if a tree contains no node of rank 1, then the number of leaves is larger than the number of internal nodes. Since M_i' contains no nullary or unary constant, we have $\#_{\mathscr{C}_0^-}(M_i) < n_i$. Thus the standard β-expansion is good. □

Lemma 4.29. *If M has a good β-expansion and contains a nonlexical constant, then M contains some lexical constant.*

Proof. Let $(\lambda x_1 \ldots x_m.M_0)M_1 \ldots M_m$ be a standard β-expansion of M. Since M contains a nonlexical constant, there is M_k with $\#_{\mathscr{C}_0^-}(M_k) \geq 1$. By $\#_{\mathscr{C}_0^-}(M_k) \leq \#(M_0, x_k)$, M_0 contains a lexical constant. □

Lemma 4.30. *For a semilexicalized ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s\rangle \in \mathbf{G}(m, n)$ which contains no nullary or unary nonlexical constants, let $\mathscr{G}' = \langle \Sigma_0', \Sigma_0, \mathscr{L}', s\rangle \in \mathbf{G}(m, 1)$ where Σ_0' and $\mathscr{L}'$ are defined in Definition 4.25. We have $\mathcal{A}(\mathscr{G}) = \mathcal{O}(\mathscr{G}')$.*

Proof. The inclusion $\mathcal{O}(\mathscr{G}') \subseteq \mathcal{A}(\mathscr{G})$ is trivial.

To show the converse, by induction on the size of M, we show that if a β-normal term $M \in \Lambda(\Sigma_0)$ has a good β-expansion, then there is $P \in \Lambda(\Sigma_0')$ such that $\tau_0'(P) = \tau_0(M)$ and $\mathscr{L}'(P) = M$.

Let $(\lambda\vec{x}.M_0)\vec{M}$ be a good β-expansion of M. There are four cases, depending on the form of M_0. Without loss of generality, we can assume that every element of $\vec{M}$ contains a constant. If $(\lambda\vec{x}_1 x_0 \vec{x}_2.M_0)\vec{M}_1 L \vec{M}_2$ is a good β-expansion of M where L is constant-free, then $(\lambda\vec{x}_1\vec{x}_2.M_0[L/x_0])\vec{M}_1\vec{M}_2$ is also a good β-expansion.

Case 1. $M_0 = \lambda z.M_0'$. Clearly $(\lambda\vec{x}.M_0')\vec{M}$ is a good β-expansion of its β-normal form $M' = |(\lambda\vec{x}.M_0')\vec{M}|_\beta$. Since M' is strictly smaller than $M = \lambda z.M'$, by the induction hypothesis, we get $P' \in \Lambda(\Sigma_0')$ with $\tau_0'(P') = \tau_0(M')$ and $\mathscr{L}'(P') = M'$. For $P = \lambda z.P' \in \Lambda(\Sigma_0')$, we have $\tau_0'(P) = \tau_0(M)$ and $\mathscr{L}'(P) = M$.

Case 2. $M_0 = wN_{1,0}\dots N_{n,0}$ for some variable w not in $\vec{x}$. Let $\vec{x}_i$ be the subsequence of $\vec{x}$ such that $\vec{x}_i$ consists of the variables appearing in $N_{i,0}$, and $\vec{M}_i$ the corresponding subsequence of $\vec{M}$. Let $N_i = N_{i,0}[\vec{M}_i/\vec{x}_i]$. That is,

$$\begin{aligned} M = M_0[\vec{M}/\vec{x}] &= wN_{1,0}\dots N_{n,0}[\vec{M}/\vec{x}] \\ &= w(N_{1,0}[\vec{M}_1/\vec{x}_1])\dots(N_{n,0}[\vec{M}_n/\vec{x}_n]) = wN_1\dots N_n. \end{aligned}$$

Clearly $(\lambda\vec{x}_i.N_{i,0})\vec{M}_i$ is a good β-expansion of N_i (see Remark 4.11). By the induction hypothesis, we have $P_i \in \Lambda(\Sigma_0')$ with $\tau_0'(P_i) = \tau_0(N_i)$ and $\mathscr{L}'(P_i) = N_i$. For $P = wP_1\dots P_n \in \Lambda(\Sigma_0')$, we have $\tau_0'(P) = \tau_0(M)$ and $\mathscr{L}'(P) = M$.

Case 3. $M_0 = \mathsf{a}N_{1,0}\dots N_{n,0}$ for some lexical constant $\mathsf{a} \in \mathscr{C}_0^+$ of type $\gamma_1 \to \dots \to \gamma_n \to p$. Let $x_{i,1}\dots x_{i,m_i}$ be the subsequence of $\vec{x}$ such that $x_{i,j}$ appears in $N_{i,0}$, and $M_{i,1}\dots M_{i,m_i}$ the corresponding subsequence of $\vec{M}$. Let

$$N_i = N_{i,0}[M_{i,j}/x_{i,j}]_{1\le j\le m_i}. \tag{4.6}$$

That is,

$$\begin{aligned} (\lambda\vec{x}.M_0)\vec{M} &= (\lambda\langle x_{i,j}\rangle_{1\le j\le m_i}^{1\le i\le n}.M_0)\langle M_{i,j}\rangle_{1\le j\le m_i}^{1\le i\le n} \\ &= M_0[M_{i,j}/x_{i,j}]_{1\le j\le m_i}^{1\le i\le n} \\ &= \mathsf{a}N_{1,0}\dots N_{n,0}[M_{i,j}/x_{i,j}]_{1\le j\le m_i}^{1\le i\le n} \\ &= \mathsf{a}(N_{1,0}[M_{1,j}/x_{1,j}]_{1\le j\le m_1})\dots(N_{n,0}[M_{n,j}/x_{n,j}]_{1\le j\le m_n}) \\ &= \mathsf{a}N_1\dots N_n = M. \end{aligned}$$

For each i, j with $1 \le i \le n$ and $1 \le j \le m_i$, Let

$$d_{i,j} = \#(M_0, x_{i,j}) - \#(N_{i,0}, x_{i,j}), \tag{4.7}$$

that is, $d_{i,j}$ denotes the number of paths from negative occurrences of atomic types in the type of the variable $x_{i,j}$ ending in the type of the head occurrence of a in M_0. Hence $\sum_{1\le j\le m_i} d_{i,j} \le |\gamma_i|$ holds. Let

$$k_{ij} = \min\{d_{i,j},\, \#_{\mathscr{C}_0^-}(M_{i,j})\}. \tag{4.8}$$

We have

$$\sum_{1\leq j\leq m_i} k_{ij} \leq \sum_{1\leq j\leq m_i} d_{i,j} \leq |\gamma_i|. \tag{4.9}$$

Let $M'_{i,j}$ be obtained from $M_{i,j}$ by replacing first k_{ij} occurrences of nonlexical constants $\mathsf{c}_{i,j,1}, \ldots, \mathsf{c}_{i,j,k_{ij}}$ with fresh variables $y_{i,j,1}, \ldots, y_{i,j,k_{ij}}$, i.e.,

$$M_{i,j} = M'_{i,j}[\mathsf{c}_{i,j,1}/y_{i,j,1}, \ldots, \mathsf{c}_{i,j,k_{ij}}/y_{i,j,k_{ij}}].$$

Let

$$N'_i \equiv N_{i,0}[M'_{i,j}/x_{i,j}]_{1\leq j\leq m_i},$$

i.e., by (4.6),

$$N_i \equiv N'_i[\vec{\mathsf{c}}_i/\vec{y}_i], \tag{4.10}$$

where $\vec{y}_i = \langle y_{i,j,h}\rangle_{1\leq h\leq k_{ij}}^{1\leq j\leq m_i}$ and $\vec{\mathsf{c}}_i = \langle \mathsf{c}_{i,j,h}\rangle_{1\leq h\leq k_{ij}}^{1\leq j\leq m_i}$. Since $(\lambda\vec{x}.M_0)\vec{M}$ is a good β-expansion of M, $\#_{\mathscr{C}_0^-}(M_{i,j}) - \#(M_0, x_{i,j}) \leq 0$ holds. Hence

$$\begin{aligned}
\#_{\mathscr{C}_0^-}(M'_{i,j}) &= \#_{\mathscr{C}_0^-}(M_{i,j}) - k_{ij} \\
&= \max\{\, 0,\ \#_{\mathscr{C}_0^-}(M_{i,j}) - d_{i,j} \,\} && \text{(by (4.8))} \\
&= \max\{\, 0,\ \#_{\mathscr{C}_0^-}(M_{i,j}) - (\#(M_0, x_{i,j}) - \#(N_{i,0}, x_{i,j})) \,\} \\
& && \text{(by (4.7))} \\
&\leq \max\{\, 0,\ \#(N_{i,0}, x_{i,j}) \,\} \\
&= \#(N_{i,0}, x_{i,j}).
\end{aligned}$$

Therefore, $(\lambda x_{i,1} \ldots x_{i,m_i}.N_{i,0})M'_{i,1} \ldots M'_{i,m_i}$ is a good β-expansion of N'_i. By the induction hypothesis, we have $P'_i \in \Lambda(\Sigma'_0)$ such that

$$\mathscr{L}'(P'_i) = N'_i. \tag{4.11}$$

Since $|\vec{\mathsf{c}}_i| = \sum_{1\leq j\leq m_i} k_{ij} \leq |\gamma_i|$ by (4.9), we have a constant $\mathsf{A} \in \mathscr{C}'_0$ such that

$$\mathscr{L}'(\mathsf{A}) \equiv \lambda w_1 \ldots w_n.\mathsf{a}(w_1\vec{\mathsf{c}}_1)\ldots(w_n\vec{\mathsf{c}}_n).$$

For $P \equiv \mathsf{A}(\lambda\vec{y}_1.P'_1)\ldots(\lambda\vec{y}_n.P'_n) \in \Lambda(\Sigma'_0)$, we have

$$\begin{aligned}
\mathscr{L}'(P) &= \mathscr{L}'(\mathsf{A})(\lambda\vec{y}_1.N'_1)\ldots(\lambda\vec{y}_n.N'_n) && \text{(by (4.11))} \\
&= \mathsf{a}(N'_1[\vec{\mathsf{c}}_1/\vec{y}_1])\ldots(N'_n[\vec{\mathsf{c}}_n/\vec{y}_n]) \\
&= \mathsf{a}N_1 \ldots N_n && \text{(by (4.10))} \\
&= M.
\end{aligned}$$

Case 4. For a good β-expansion $(\lambda\vec{x}.M_0)\vec{M}$ of M, M_0 is of the form $x_i\vec{K}$ for some x_i in $\vec{x}$. Without loss of generality, we can assume that it is x_1. Let

$$\begin{aligned} M_0 &= x_1 K_1 \dots K_m \\ M &= M_0[\vec{M}/\vec{x}] \\ &= x_1 K_1 \dots K_m[M_1/x_1, \vec{M}_1/\vec{x}_1, \dots, \vec{M}_m/\vec{x}_m] \\ &= M_1 K_1[\vec{M}_1/\vec{x}_1] \dots K_m[\vec{M}_m/\vec{x}_m] \end{aligned}$$

where $\vec{x}_i$ is the subsequence of $\vec{x}$ consisting of the variables in K_i, and $\vec{M}_i$ is the corresponding subsequence of $\vec{M}$. Since x_1 has a second-order type, each K_i has an atomic type.

Recall that we assume that M_1 contains at least one nonlexical constant. Since

$$\#(M_0, x_1) \geq \#_{\mathscr{C}_0^-}(M_1) \geq 1,$$

there is K_k for $k \in \{1, \dots, m\}$ of the form

$$K_k = \mathsf{a} N_{1,0} \dots N_{n,0}$$

for some $\mathsf{a} \in \mathscr{C}_0^+$ of type $\gamma_1 \to \dots \to \gamma_n \to p$ and $N_{i,0} \in \Lambda(\Sigma_0^+)$ of type γ_i for $i \in \{1, \dots, n\}$. Let M_1' be obtained from M_1 by replacing an occurrence of a constant $\mathsf{c}_0 \in \mathscr{C}_0^-$ with y_0, and M_0' obtained from M_0 by replacing K_k with a fresh variable z_k, i.e.,

$$\begin{aligned} M_1 &= M_1'[\mathsf{c}_0/y_0] \\ M_0' &= x_1 K_1 \dots K_{k-1} z_k K_{k+1} \dots K_m \end{aligned}$$

and define M' as

$$M' = M_0'[M_1'/x_1, \vec{M}_1/\vec{x}_1, \dots, \vec{M}_{k-1}/\vec{x}_{k-1}, \vec{M}_{k+1}/\vec{x}_{k+1}, \dots, \vec{M}_m/\vec{x}_m]$$

Then,

$$\begin{aligned} M &= M_0[M_1/x_1, \vec{M}_1/\vec{x}_1, \dots, \vec{M}_m/\vec{x}_m] \\ &= M_0'[K_k/z_k][M_1'[\mathsf{c}_0/y_0]/x_1, \vec{M}_1/\vec{x}_1, \dots, \vec{M}_m/\vec{x}_m] \\ &= M'[\mathsf{c}_0/y_0, K_k[\vec{M}_k/\vec{x}_k]/z_k]. \end{aligned} \tag{4.12}$$

For $i \neq k$, we have $\#(M_0', x_{i,j}) = \#(M_0, x_{i,j}) \geq \#_{\mathscr{C}_0^-}(M_{i,j})$ for every $x_{i,j}$ in $\vec{x}_i$ and the corresponding term $M_{i,j}$ in $\vec{M}_i$, and

$$\#_{\mathscr{C}_0^-}(M_1') = \#_{\mathscr{C}_0^-}(M_1) - 1 \leq \#(M_0, x_1) - 1 = \#(M_0', x_1).$$

Therefore

$$(\lambda x_1 \vec{x}_1 \dots \vec{x}_{k-1} \vec{x}_{k+1} \dots \vec{x}_m . M'_0) M'_1 \vec{M}_1 \dots \vec{M}_{k-1} \vec{M}_{k+1} \dots \vec{M}_m$$

is a good β-expansion of M'. By the induction hypothesis we have $P' \in \Lambda(\Sigma'_0)$ such that

$$\mathscr{L}'(P') = M'.$$

On the other hand, by $\#(M_0, x_{i,j}) = \#(K_i, x_{i,j})$ for each $x_{i,j}$ in $\vec{x}_i$ for all $i \in \{1, \dots, m\}$, $(\lambda \vec{x}_i . K_i)\vec{M}_i$ is a good β-expansion of $K_i[\vec{M}_i/\vec{x}_i]$. Therefore, we can apply the same discussion[5] in Case 3 to

$$K_k[\vec{M}_k/\vec{x}_k] = \mathsf{a} N_{1,0} \dots N_{n,0}[\vec{M}_k/\vec{x}_k],$$

and then we find $P'_i \in \Lambda(\Sigma'_0)$ and a sequence $\vec{\mathsf{c}}_i$ of nonlexical constants such that $|\vec{\mathsf{c}}_i| \leq |\gamma_i|$ and

$$\mathscr{L}'(P'_i)[\vec{\mathsf{c}}_i/\vec{y}_i] = N_{i,0}[\vec{M}_k/\vec{x}_k].$$

By the definition of Σ'_0, we have a constant B which is mapped to

$$\mathscr{L}'(\mathsf{B}) = \lambda w_0 \dots w_n . w_0 \mathsf{c}_0 (\mathsf{a}(w_1 \vec{\mathsf{c}}_1) \dots (w_n \vec{\mathsf{c}}_n)).$$

Thus, for $P = \mathsf{B}(\lambda y_0 z_k . P')(\lambda \vec{y}_1 . P'_1) \dots (\lambda \vec{y}_n . P'_n) \in \Lambda(\Sigma'_0)$, we have

$$\begin{aligned}
\mathscr{L}'(P) &= \big(\lambda y_0 z_k . \mathscr{L}'(P')\big) \mathsf{c}_0 \big(\mathsf{a}(\mathscr{L}'(P'_1)[\vec{\mathsf{c}}_1/\vec{y}_1]) \dots (\mathscr{L}'(P'_n)[\vec{\mathsf{c}}_n/\vec{y}_n])\big) \\
&= M'[\mathsf{c}_0/y_0, \mathsf{a} N_{i,0} \dots N_{n,0}[\vec{M}_k/\vec{x}_k]/z_k] \\
&= M'[\mathsf{c}_0/y_0, K_k[\vec{M}_k/\vec{x}_k]/z_k] \\
&= M. \qquad \text{(by (4.12))}
\end{aligned}$$

□

Theorem 4.31. *For every semilexicalized ACG $\mathscr{G} \in \mathbf{G}(m, n)$, there is a lexicalized ACG $\mathscr{G}' \in \mathbf{G}(\max\{4, m\}, n)$ such that*

$$\mathcal{O}(\mathscr{G}') = \{\, R \in \mathcal{O}(\mathscr{G}) \mid R \text{ contains a constant} \,\}.$$

Proof. By Lemmas 4.28 and 4.30. □

[5] Substitute $K_k[\vec{M}_k/\vec{x}_k]$ for M in Case 3. K_k in Case 4 and M_0 in Case 3 have the same form.

4.3.5 Third-Order Case

In the previous subsection, for the fourth or higher-order case, we have constructed a lexicalized ACG $\mathscr{G}^l = \langle \Sigma'_0, \Sigma_1, \mathscr{L} \circ \mathscr{L}', s \rangle$, where the ACG $\mathscr{G}' = \langle \Sigma'_0, \Sigma_0, \mathscr{L}', s \rangle$ "controls" nonlexical (w.r.t. $\mathscr{L}$) constants $\mathsf{c} \in \mathscr{C}_0^-$ by bound variables in its lexical entries, and then we can put nonlexical constants in almost arbitrary places. Since variables that can be functors of second-order nonlexical constants are at least third-order, then the order of abstract vocabulary becomes more than three.

Recall Example 4.26. For the third-order semilexicalized ACG $\mathscr{G} \in \mathbf{G}(3, n)$, let

$$M = \mathsf{a}_1(\lambda x_1^s.\mathsf{a}_2(\lambda x_2^s.\mathsf{a}_3(\lambda x_3^s.\mathsf{a}_4(\lambda x_4^s.\mathsf{c}_1(\mathsf{c}_2 x_1 x_2)(\mathsf{c}_3 x_3 x_4))))) \in \mathcal{A}(\mathscr{G}) \tag{4.13}$$

where $\mathsf{a}_i \in \mathscr{C}_0^+$ has type $(s \to s) \to s$ and $\mathsf{c}_i \in \mathscr{C}_0^-$ has type $s^2 \to s$. Here we number each occurrence of a constant to facilitate discussion. In Example 4.26, we first replace occurrences of c_i with new bound variables y_i, and then replace occurrences of $\mathsf{a}_i N_i$ with $\mathsf{A}_i(\lambda y_i.N_i)$, where $\mathsf{A}_i \in \mathscr{C}'_0$ is a new abstract constant mapped to $\lambda w.\mathsf{a}_i(w\mathsf{c}_i)$. The resultant term is

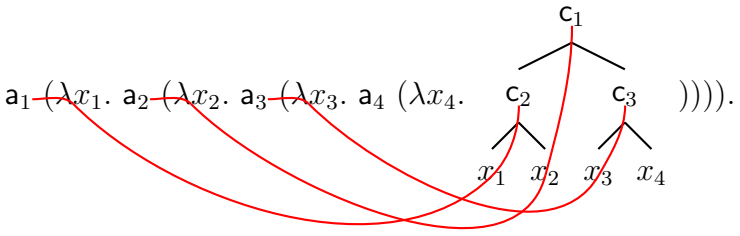
$$\mathsf{A}_1(\lambda y_1^{s^2 \to s} x_1^s.\mathsf{A}_2(\lambda y_2^{s^2 \to s} x_2^s.\mathsf{A}_3(\lambda y_3^{s^2 \to s} x_3^s.\mathsf{a}_4(\lambda x_4^s.y_1(y_2 x_1 x_2)(y_3 x_3 x_4))))). \tag{4.14}$$

This way we associate occurrences of nonlexical constants with occurrences of lexical constants. To keep the order of the abstract vocabulary in three, however, that strategy must be abandoned. We are no longer able to use second-order variables in the abstract vocabulary.

Recall the idea that we have used for lexicalization of second-order ACGs, where the third-order lexicalized ACG $\mathscr{G}'$ in Proposition 4.18 is transformed into the second-order lexicalized ACG $\mathscr{G}^l$ in Definition 4.20. We have introduced new abstract atomic types in $\mathscr{A}'_0$ mapped to second-order types on $\mathcal{T}(\mathscr{A}_0)$ by the new lexicon $\mathscr{L}'$. That trick works well because the usage of free variables and λ-abstraction in an element of $\mathcal{A}(\mathscr{G}')$ is very much limited so that we can embed terms involving λ-abstraction into finitely many lexical entries of $\mathscr{G}''$ shown in (4.5) on page 64.

We want to apply that technique to the third-order case. As we see in (4.14), the lexicalization method for the fourth or higher-order case constructs the form of the nonlexical part $\mathsf{c}_1(\mathsf{c}_2 x_1 x_2)(\mathsf{c}_3 x_3 x_4)$ by variables y_i of second-order type, and then substitutes nonlexical constants for those variables. To exclude variables of second-order types from terms in the abstract language in the third-order case, we must embed those variables into the lexical entries so that new lexical entries can build any nonlexical parts in a term in the abstract language.

Let us associate occurrences of nonlexical constants with ones of lexical constants in M as follows:



According to this association, we let an ACG $\mathscr{G}' = \langle \Sigma'_0, \Sigma_0, \mathscr{L}', [s] \rangle \in \mathbf{G}(3,2)$ consist of the following lexical entries.[6]

$\mathsf{x} \in \mathscr{C}'_0$	$\tau'_0(\mathsf{x})$	$\mathscr{L}'(\mathsf{x})$
a_4	$[s \to s] \to [s]$	a_4
A_3	$[s \to s] \to ([s \to s] \to [s]) \to [s]$	$\lambda yw.\mathsf{a}_3(\lambda x_3.w(\lambda x_4.y(\mathsf{c}_3 x_3 x_4)))$
A_2	$[s \to s] \to ([s \to s] \to [s]) \to [s]$	$\lambda yw.\mathsf{a}_2(\lambda x_2.w(\lambda v.\mathsf{c}_1(yx_2)v))$
A_1	$([s \to s] \to [s]) \to [s]$	$\lambda w.\mathsf{a}_1(\lambda x_1.w(\lambda v.\mathsf{c}_2 x_1 v))$

where $[s \to s]$ and $[s]$ are new abstract atomic types mapped to $s \to s$ and s, respectively. It is a routine to check that $\mathscr{L}'(N) = M$ for

$$N = \mathsf{A}_1(\lambda y_1^{[s \to s]}.\mathsf{A}_2 y_1(\lambda y_2^{[s \to s]}.\mathsf{A}_3 y_2 \mathsf{a}_4)).$$

Elimination of Unary Nonlexical Constants

We start lexicalization of third-order semilexicalized ACGs with a modification of Lemma 4.24. While Lemmas 4.16 and 4.23 (elimination of nullary nonlexical constants) are good enough for third-order ACGs, Lemma 4.24 is not.

Lemma 4.32 (Elimination of Unary Nonlexical Constants). *For every semilexicalized ACG $\mathscr{G} \in \mathbf{G}(3,n)$, we can find a semilexicalized ACG $\mathscr{G}' \in \mathbf{G}(3,n)$ such that $\mathscr{G}'$ contains no nullary or unary nonlexical constants and*

$$\mathcal{O}(\mathscr{G}') = \{\, R \in \mathcal{O}(\mathscr{G}) \mid R \text{ contains a constant} \,\}.$$

Proof. We assume that $\mathscr{G}$ contains no nullary nonlexical constants and satisfies Condition II in Definition 4.12. $\mathcal{O}(\mathscr{G})$ contains no combinators.

We let an ACG $\mathscr{G}'$ consist of the following lexical entries:

[6] Our actual method generates a lexicalized ACG whose lexical entries are a little more complicated.

- $\langle \mathsf{c},\ \tau_0(\mathsf{c}),\ \mathscr{L}(\mathsf{c})\rangle$ for all $\mathsf{c} \in \mathscr{C}_0^-$ unless $\tau_0(\mathsf{c})$ is unary,
- $\langle [\![M]\!],\ \tau_0(M),\ \mathscr{L}(M)\rangle$ for each M of the form

$$M = \lambda x_1 \dots x_n.\mathsf{c}_0(\mathsf{a}(\lambda \vec{z}_1.x_1 M_{i,1} \dots M_{1,m_1}) \dots (\lambda \vec{z}_n.x_n M_{n,1} \dots M_{n,m_n}))$$

where $\vec{z}_i = z_{i,1}^{p_{i,1}} \dots z_{i,m_i}^{p_{i,m_i}}$, $\mathsf{a} \in \mathscr{C}_0^+$,

$$\tau_0(\mathsf{a}) = (p_{1,1} \to \dots \to p_{1,m_1} \to p_1) \to \dots \to (p_{n,1} \to \dots \to p_{n,m_n} \to p_n) \to p,$$

$$M_{i,j} = \begin{cases} \mathsf{c}_{i,j} z_{i,j} & \text{if } j \in I_i \\ z_{i,j} & \text{if } j \notin I_i \end{cases} \quad \text{for some } I_i \subseteq \{1, \dots, m_i\},$$

$$\mathsf{c}_0, \mathsf{c}_{i,j} \in \mathscr{C}_0^-, \quad \tau_0(\mathsf{c}_0) = p \to q, \quad \tau_0(\mathsf{c}_{i,j}) = p_{i,j} \to q_{i,j},$$
$$\tau_0(x_i) = q_{i,1} \to \dots \to q_{i,m_i} \to p_i \text{ where } p_{i,j} = q_{i,j} \text{ if } j \notin I_i.$$

Since the order of the type of the bound variables x_i in the above M is at most 2, we have $\mathrm{ord}(\tau_0'([\![M]\!])) \le \max\{\, 3,\ \mathrm{ord}(\tau_0(\mathsf{a})) \mid \mathsf{a} \in \mathscr{C}_0 \,\}$. Thus, $\mathscr{G}' \in \mathbf{G}(3, n)$ if $\mathscr{G} \in \mathbf{G}(3, n)$.

The inclusion $\mathcal{O}(\mathscr{G}') \subseteq \mathcal{O}(\mathscr{G})$ is trivial. We show the converse relation $\mathcal{O}(\mathscr{G}) \subseteq \mathcal{O}(\mathscr{G}')$.

We say that an occurrence of a unary nonlexical constant of type $p \to q$ is *linked to* an occurrence of a lexical constant if the path which starts from the negative occurrence of the subtype p in the type $p \to q$ of the occurrence of the unary nonlexical constant ends in p in the type of the occurrence of the lexical constant. We say that a term $M \in \Lambda(\Sigma_0)$ satisfies the *linking condition* if for every occurrence of a unary nonlexical constant, there is an occurrence of a lexical constant to which the occurrence of the unary nonlexical constant is linked. In fact, we can assume that every $M \in \mathcal{A}(\mathscr{G})$ satisfies the linking condition, as explained in the proof of Lemma 4.24. In the third-order case, when a unary nonlexical constant c is linked to a lexical constant a in some term in long normal form, their occurrences can be written either

- $\mathsf{c}(\mathsf{a}\vec{M})$ (direct link) or
- $\mathsf{a}\vec{M}(\lambda \vec{y} z.N[\mathsf{c}z/x])$,

since the order of a is at most three. There cannot be two or more bound variables on which a path passes such as

$$\mathsf{a}(\lambda y.y(\lambda z.\mathsf{c}z)).$$

If the above term is well-typed, the order of a must be more than three.

Now, we show by induction on the size of M that for every term $M \in \Lambda(\Sigma_0)$ satisfying the linking condition, there is $P \in \Lambda(\Sigma_0')$ such that $\tau_0'(P) = \tau_0(M)$ and $\mathscr{L}'(P) = \mathscr{L}(M)$. We assume that M is in long normal form.

If M contains no constants, let $P = M \in \Lambda(\Sigma_0')$. Otherwise, it is enough to treat cases where M is of the form

$$M = \mathsf{b}M_1 \dots M_n$$

where b is either a lexical constant or a unary nonlexical constant. Other cases are trivial as in the proof of Lemma 4.23.

Case 1. Suppose that the head of M is a unary nonlexical constant $\mathsf{c}_0 \in \mathscr{C}_0^-$. By the linking condition, M is of the form

$$M = \mathsf{c}_0(\mathsf{a}M_1 \dots M_n)$$

for some $\mathsf{a} \in \mathscr{C}_0^+$ of type

$$\tau_0(\mathsf{a}) = (p_{1,1} \to \dots \to p_{1,m_1} \to p_1) \to \dots \to (p_{n,1} \to \dots \to p_{n,m_n} \to p_n) \to p.$$

We can assume that M_i has the form $M_i = \lambda \vec{z}_i.M_i'$ with $\vec{z}_i = z_{i,1}^{p_{i,1}} \dots z_{i,m_i}^{p_{i,m_i}}$. Let

$$I_i = \{\, j \mid \text{some occurrence of a unary nonlexical constant } \mathsf{c}_{i,j} \text{ is linked to } \mathsf{a} \text{ via } p_{i,j} \,\}.$$

Then, we can find M_i'' and $M_{i,j}$ such that

$$M_i' = M_i''[M_{i,j}/y_{i,j}]_{1 \le j \le m_i}$$
$$M_{i,j} = \begin{cases} \mathsf{c}_{i,j} z_{i,j} & \text{if } j \in I_i, \\ z_{i,j} & \text{if } j \in \{1, \dots, m_i\} - I_i. \end{cases}$$

Clearly each M_i'' satisfies the linking condition. By the induction hypothesis, there is P_i' such that $\tau_0'(P_i') = \tau_0(M_i'')$ and $\mathscr{L}'(P_i') = \mathscr{L}(M_i'')$. By the definition of $\mathscr{G}'$, there is a constant $[\![N]\!] \in \mathscr{C}_0'$ for

$$N = \lambda x_1 \dots x_n.\mathsf{c}_0(\mathsf{a}(\lambda \vec{z}_1.x_1 M_{1,1} \dots M_{1,m_1}) \dots (\lambda \vec{z}_n.x_n M_{n,1} \dots M_{n,m_n})).$$

Thus, for

$$P = [\![N]\!](\lambda y_{1,1} \dots y_{1,m_1}.P_1') \dots (\lambda y_{n,1} \dots y_{n,m_n}.P_n'),$$

we have

$$\begin{aligned}\mathscr{L}'(P) &= \mathscr{L}(N)(\lambda y_{1,1} \dots y_{1,m_1}.\mathscr{L}(M_1'')) \dots (\lambda y_{n,1} \dots y_{n,m_n}.\mathscr{L}(M_n'')) \\ &= \mathscr{L}\big(\mathsf{c}_0(\mathsf{a}(\lambda \vec{z}_1.M_1'' M_{1,1} \dots M_{1,m_1}) \dots (\lambda \vec{z}_n.M_n'' M_{n,1} \dots M_{n,m_n}))\big) \\ &= \mathscr{L}\big(\mathsf{c}_0(\mathsf{a}(\lambda \vec{z}_1.M_1') \dots (\lambda \vec{z}_n.M_n'))\big) \\ &= \mathscr{L}(M).\end{aligned}$$

Case 2. Suppose that the head of M is a lexical constant $\mathsf{a} \in \mathscr{C}_0^+$ and M is of the form $M = \mathsf{a}M_1 \dots M_n$. Since M is long normal form, M has an atomic type p. Since $\mathscr{G}'$ satisfies Condition II, there is $\mathsf{e}_p \in \mathscr{C}_0$ such that $\tau_0(\mathsf{e}_p) = p \to p$ and $\mathscr{L}(\mathsf{e}_p) = \lambda x.x$. By $\mathsf{e}_p M \approx M$, this case reduces to Case 1, where the induction hypothesis is applied to terms strictly smaller than M. □

Technical Definition and Lemma

As a preparation for elimination of nonlexical constants, we need some technical definition and lemma. These are necessary for building nonlexical parts in a term in the original abstract language. The reader can assume that the following second-order signature Σ is the nonlexical part Σ_0^- of the abstract vocabulary of the given semilexicalized ACG.

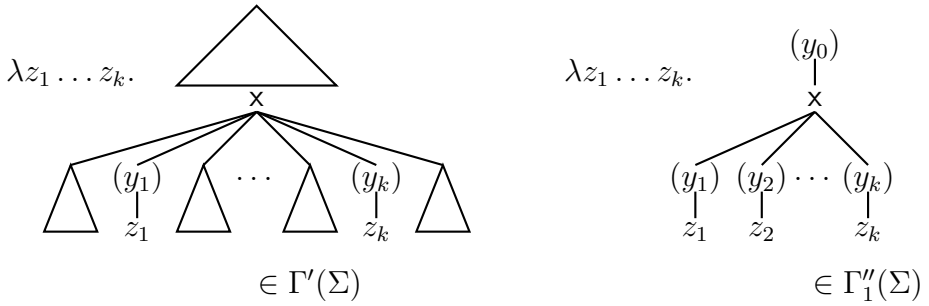
Hereafter we denote the set of nullary free variables of M by $\mathrm{Fv}^0(M)$. We say that a second-order type is *multi-ary* if it is k-ary for $k \geq 2$. Suppose that a second-order signature $\Sigma = \langle \mathscr{A}, \mathscr{C}, \tau \rangle$ is such that every constant has a multi-ary type. Let $k_\Sigma = \max\{\, k \mid \tau_0(\mathsf{c}) = p_1 \to \dots \to p_k \to q, \mathsf{c} \in \mathscr{C} \,\}$. We define a subset $\Gamma(\Sigma)$ of $\Lambda(\Sigma)$ to consist of λ-terms M such that

- M has an atomic type,
- M is built from variables of second-order types whose arities are strictly less than k_Σ and constants of Σ,
- there is no subterm of the form $y_1(y_2 M'))$ for any two unary variables y_1 and y_2.

Formally $\Gamma(\Sigma)$ is defined by

$$\begin{aligned}
\Gamma_0^-(\Sigma) &= \{\, z^p \mid p \in \mathscr{A} \,\},\\
\Gamma_n(\Sigma) &= \Gamma_n^-(\Sigma) \cup \{\, y^{p\to q} M \mid M \in \Gamma_n^-(\Sigma) \text{ and } \tau(M) = p \,\},\\
\Gamma_{n+1}^-(\Sigma) &= \Gamma_n^-(\Sigma) \cup \{\, \mathsf{c} N_1 \dots N_k \mid N_1, \dots, N_k \in \Gamma_n(\Sigma),\\
&\qquad\qquad \tau_0(\mathsf{c}) = \tau(N_1) \to \dots \to \tau(N_k) \to q \,\}\\
&\quad \cup \{\, x^{\tau(N_1)\to\dots\to\tau(N_k)\to q} N_1 \dots N_k \mid N_1, \dots, N_k \in \Gamma_n(\Sigma),\ 2 \le k < k_\Sigma \,\},\\
\Gamma(\Sigma) &= \bigcup_{n\ge 0} \Gamma_n(\Sigma).
\end{aligned}$$

We denote by $\Gamma''(\Sigma)$ the set of λ-terms obtained from elements of $\Gamma(\Sigma)$ by binding all the nullary variables in them. $\Gamma_1''(\Sigma)$ is the finite subset of $\Gamma''(\Sigma)$ whose elements contain at most one occurrence of a constant or multi-ary variable. Each element of $\Gamma'(\Sigma)$ is obtained from an element of $\Gamma(\Sigma)$ by binding some nullary variables that have the same parent when we ignore unary variables, but the term rooted by that parent contains at least one nullary free variable. Note that if $M \in \Gamma'(\Sigma)$, then the arity of M is less than k_Σ.



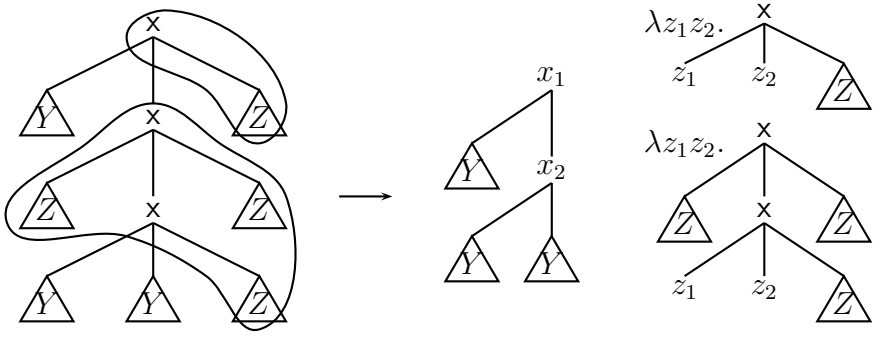
$\in \Gamma'(\Sigma)$ $\qquad\qquad$ $\in \Gamma_1''(\Sigma)$

Formally each of the above sets are defined as follows:

$$\begin{aligned}
\Gamma''(\Sigma) &= \{\, \lambda\vec{z}.N \mid N \in \Gamma(\Sigma),\ \vec{z} \text{ is a sequence of all the nullary variables in } N \,\},\\
\Gamma_1''(\Sigma) &= \{\, \lambda\vec{z}.M \in \Gamma''(\Sigma) \mid M \in \Gamma_1(\Sigma) \,\},\\
\Gamma'(\Sigma) &= \{\, \lambda z_1 \dots z_k.M \mid M \in \Gamma(\Sigma),\ M = M'[\mathsf{x}\vec{M}_0 N_1 \vec{M}_1 \dots N_k \vec{M}_k / z],\\
&\qquad k \ge 1,\ \mathsf{x} \in \mathscr{C} \cup \mathscr{X},\ \vec{M}_0, \dots, \vec{M}_k \in \Gamma(\Sigma),\ N_1, \dots, N_k \in \Gamma_0(\Sigma),\\
&\qquad \mathrm{Fv}^0(N_i) = \{z_i\},\ \vec{M}_0 \dots \vec{M}_k \ne \varepsilon \,\}.
\end{aligned}$$

Lemma 4.33. *For a fixed λ-term $M \in \Gamma(\Sigma)$, let us partition the set $\mathrm{Fv}^0(M)$ of nullary free variables of M into two disjoint subsets Y and Z such that $Y \ne \varnothing$. We can find $M' \in \Gamma(\Sigma)$ and $M_1, \dots, M_m \in \Gamma'(\Sigma)$ such that*

- $M = M'[M_i/x_i]_{1\leq i\leq m}$,
- $\mathrm{Fv}^0(M') = Y$ *and* $\bigcup_{1\leq i\leq m} \mathrm{Fv}^0(M_i) = Z$.



Proof. Induction on M. If $M \in \Gamma_0(\Sigma)$, then $M = yz$ or $M = z$. Since Y is not empty, $Y = \{z\}$ and $Z = \varnothing$. Letting $M' = M$, we see that the lemma holds.

Otherwise, M is of the form

$$M = \begin{cases} \mathsf{x} N_1 \dots N_n & \text{or} \\ y(\mathsf{x} N_1 \dots N_n) \end{cases}$$

where $2 \leq n \leq k_\Sigma$, x is a constant or a variable, $N_i \in \Gamma(\Sigma)$, and y is a unary variable. Let N_0 be such that

$$M = N_0 N_1 \dots N_n$$

($N_0 = \mathsf{x}$ or $N_0 = \lambda z_1 \dots z_n . y(\mathsf{x} z_1 \dots z_n)$ depending on the form of M). Let us partition $\{1, \dots, n\}$ into two subsets I and J so that

$$I = \{\, i \mid 1 \leq i \leq n,\ Y \cap \mathrm{Fv}^0(N_i) \neq \varnothing \,\}$$
$$J = \{\, i \mid 1 \leq i \leq n,\ \mathrm{Fv}^0(N_i) \subseteq Z \,\}.$$

For each $i \in I$, by applying the induction hypothesis to N_i, we get $N_i' \in \Gamma(\Sigma)$ and $N_{i,1}, \dots, N_{i,m_i} \in \Gamma'(\Sigma)$ such that

- $N_i = N_i'[N_{i,j}/x_{i,j}]_{1\leq j\leq m_i}$,
- $Y \cap \mathrm{Fv}^0(N_i) = \mathrm{Fv}^0(N_i')$ and $Z \cap \mathrm{Fv}^0(N_i) = \bigcup_{1\leq j\leq m_i} \mathrm{Fv}^0(N_{i,j})$.

We see $I \neq \varnothing$ by $Y \neq \varnothing$.

Case 1. $|I| = 1$. Let k be the unique integer such that $I = \{k\}$. Depending on the form of $N_k' \in \Gamma(\Sigma)$, we have two subcases.

Case 1.1. The head of N'_k is $x_{k,j}$ for some $j \in \{1, \ldots, m_k\}$. $N_{k,j} \in \Gamma'(\Sigma)$ can be represented as

$$N_{k,j} = \lambda \vec{z}_{k,j}.N'_{k,j}$$

for some $N'_{k,j} \in \Gamma(\Sigma)$. Let

$$M' = N'_k$$
$$M_i = \begin{cases} N_{k,i} & \text{if } i \neq j, \\ \lambda \vec{z}_{k,j}.N_0 N_1 \ldots N_{k-1} N'_{k,j} N_{k+1} \ldots N_n & \text{if } i = j \end{cases}$$

for $1 \leq i \leq m_k$. Then, we have

- $M' \in \Gamma(\Sigma)$ and $M_i \in \Gamma'(\Sigma)$ for $1 \leq i \leq m_k$,
- $M = M'[M_i/x_{k,i}]_{1 \leq i \leq m_k}$,
- $\mathrm{Fv}^0(M') = Y$ and $\bigcup_{1 \leq i \leq m_k} \mathrm{Fv}^0(M_i) = Z$.

Case 1.2. The head of N'_k is not $x_{k,j}$ for any $j \in \{1, \ldots, m_k\}$. Let

$$N''_k = \begin{cases} N''_k & \text{if } N'_k \text{ is of the form } N'_k = y' N''_k \text{ for some unary variable } y' \\ N'_k & \text{if the head of } N'_k \text{ is not a unary variable} \end{cases}$$
$$K = \begin{cases} y'z & \text{if } N'_k \text{ is of the form } N'_k = y' N''_k \text{ for some unary variable } y' \\ z & \text{if the head of } N'_k \text{ is not a unary variable.} \end{cases}$$

Then, $N'_k = K[N''_k/z]$ and the head of N''_k is not a unary variable. Let

$$M' = y_0 N''_k,$$
$$M_0 = \lambda z.N_0 N_1 \ldots N_{k-1} K N_{k+1} \ldots N_n,$$

where y_0 is a fresh unary variable of type $\tau(N''_k) \to \tau(M)$. We have

- $M' \in \Gamma(\Sigma)$ and $M_0 \in \Gamma'(\Sigma)$,
- $M = M'[M_0/y_0][N_{k,i}/x_{k,i}]_{1 \leq i \leq m_k}$,
- $\mathrm{Fv}^0(M') = Y$ and $\mathrm{Fv}^0(M_0) \cup \bigcup_{1 \leq i \leq m_k} \mathrm{Fv}^0(N_{k,i}) = Z$.

Case 2. $1 < |I| < n$. Let

$$M' = x_0 \vec{N}'_I$$
$$M_0 = \lambda \vec{z}_I.N_0 N''_1 \ldots N''_n$$
$$N''_i = \begin{cases} z_i & \text{if } i \in I \\ N_i & \text{if } i \notin I \end{cases}$$

where $\vec{N}'_I = \langle\, N'_i \mid i \in I \,\rangle$, $\vec{z}_I = \langle\, z_i \mid i \in I \,\rangle$, and $\tau(x_0) = \tau(M_0) = \tau(\vec{N}'_I) \to \tau(M)$.

Note that since $1 < |I| < n \leq k_\Sigma$, x_0 has neither a unary nor k_Σ-ary type. Therefore,

- $M' \in \Gamma(\Sigma)$ and $M_0 \in \Gamma'(\Sigma)$,
- $M = M'[M_0/x_0][N_{i,j}/x_{i,j}]^{i\in I}_{1\leq j\leq m_i}$,
- $\mathrm{Fv}^0(M') = Y$ and $\mathrm{Fv}^0(M_0) \cup \bigcup_{\substack{i\in I\\ 1\leq j\leq m_i}} \mathrm{Fv}^0(N_{i,j}) = Z$.

Case 3. $|I| = n$. Let $M' = N_0 N'_1 \dots N'_n \in \Gamma(\Sigma)$. We have

- $M = M'[N_{i,j}/x_{i,j}]^{1\leq i\leq n}_{1\leq j\leq m_i}$,
- $\mathrm{Fv}^0(M') = Y$ and $\bigcup_{\substack{1\leq i\leq n\\ 1\leq j\leq m_i}} \mathrm{Fv}^0(N_{i,j}) = Z$.

□

Elimination of Nonlexical Constants

Definition 4.34. Let $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s\rangle \in \mathbf{G}(3, n)$ be a third-order semilexicalized ACG that has no nullary or unary nonlexical constants. Let

$$\mathscr{A}'_0 = \{\, [\gamma] \mid \gamma \text{ is a second-order type with } |\gamma| \leq k_{\Sigma_0^-} \,\}$$
$$\text{where } k_{\Sigma_0^-} = \max\{\, |\tau_0(\mathsf{c})| - 1 \mid \mathsf{c} \in \mathscr{C}_0^- \,\}.$$

Let $[\vec{\alpha}] \to \beta$ denote $[\alpha_1] \to \dots \to [\alpha_m] \to \beta$ if $\vec{\alpha}$ represents the sequence $\alpha_1 \dots \alpha_m$ of types in $\mathcal{T}(\mathscr{A}_0)$ and $[\alpha_i] \in \mathscr{A}'_0$. Let $\mathscr{C}'_0$ be the set of constants $\mathsf{A}_{K,M}$ for all K and M such that

- KM is well-typed,
- $K \in \Gamma''_1(\Sigma_0^-)$,
- $M = \mathsf{a}(\lambda\vec{v}_1.w_1\vec{M}_1\vec{M}'_1)\dots(\lambda\vec{v}_m.w_m\vec{M}_m\vec{M}'_m)$,
  - $\mathsf{a} \in \mathscr{C}_0^+$ of type $(\vec{p}_1 \to p_1) \to \dots \to (\vec{p}_m \to p_m) \to p$,
  - $\vec{M}_i$ consists of elements of $\Gamma(\Sigma_0^-)$,
  - $\vec{M}'_i$ consists of elements of $\Gamma'(\Sigma_0^-)$,
  - $\vec{v}_i$ is a sequence of the nullary free variables in $\vec{M}_i$ and $\vec{M}'_i$,
  - $\tau_0(\vec{v}_i) = \vec{p}_i$,

– $\tau_0(w_i) = \tau_0(\vec{M}_i) \to \tau_0(\vec{M}'_i) \to p_i$.

Let $\{y_1, \ldots, y_l\} = \mathrm{Fv}(KM) - \{w_1, \ldots, w_m\}$. We define τ'_0 and $\mathscr{L}'$ as

- $\tau'_0(\mathsf{A}_{K,M}) = [\tau_0(y_1)] \to \cdots \to [\tau_0(y_l)] \to \gamma_1 \to \cdots \to \gamma_m \to [\tau_0(KM)]$, where $\gamma_i = [\tau_0(\vec{M}_i)] \to [\tau_0(\vec{M}'_i)] \to [p_i]$,
- $\mathscr{L}'([\gamma]) = \gamma$ for all $[\gamma] \in \mathscr{A}'_0$,
- $\mathscr{L}'(\mathsf{A}_{K,M}) = \lambda y_1 \ldots y_l w_1 \ldots w_m . KM$.

An ACG $\mathscr{G}^l$ called *the lexicalized form of* $\mathscr{G}$ is defined as

$$\mathscr{G}^l = \langle \Sigma'_0, \Sigma_1, \mathscr{L} \circ \mathscr{L}', [s] \rangle \in \mathbf{G}(3, n+1)$$

where $\Sigma'_0 = \langle \mathscr{A}'_0, \mathscr{C}'_0, \tau'_0 \rangle$.

We need to check the well-definedness of the above definition. First we check that $\tau'_0(\mathsf{A}_{K,M}) \in \mathcal{T}(\mathscr{A}'_0)$. Since every element of $\Gamma(\Sigma_0^-)$ has an atomic type, and every element of $\Gamma'(\Sigma_0^-)$ has less than $k_{\Sigma_0^-}$-ary type, we see $\gamma_i \in \mathcal{T}(\mathscr{A}'_0)$. By the definition of $\Gamma(\Sigma_0^-)$ and $\Gamma'(\Sigma_0^-)$, the arities of variables $y_1, \ldots, y_l$ are less than $k_{\Sigma_0^-}$. Therefore, $[\tau_0(y_j)] \in \mathscr{A}'_0$ and $\tau'_0(\mathsf{A}_{K,M}) \in \mathcal{T}(\mathscr{A}'_0)$.

Second we check that $\mathscr{C}'_0$ is finite. Since $\Gamma''_1(\Sigma_0^-)$ is a finite set, finitely many possible forms of K exist. In M, the length of $\vec{v}_i$ for $1 \leq i \leq n$ is determined by the type of a. Since each element of $\vec{M}_i, \vec{M}'_i$ has at least one element of $\vec{v}_i$ and each variable in $\vec{v}_i$ appears exactly once in some element of $\vec{M}_i, \vec{M}'_i$, thus we see $|\vec{M}_i \vec{M}'_i| \leq |\vec{v}_i|$. For each $M_{i,j}$ in $\vec{M}_i$, $\mathrm{Fv}^0(M_{i,j})$ consists of some variables in $\vec{v}_i$ only. If $M_{i,j}$ contains no unary variables, then the size of $M_{i,j}$ is bounded by $|\mathrm{Fv}^0(M_{i,j})| \leq |\vec{v}_i|$, otherwise, the size of $M_{i,j}$ is bounded by $2|\mathrm{Fv}^0(M_{i,j})|$ by the restriction on occurrences of unary variables in elements of $\Gamma(\Sigma_0^-)$. Similarly, the size of $M'_{i,j}$ in $\vec{M}'_i$ is bounded by $2(|\mathrm{Fv}^0(M'_{i,j})| + k_{\Sigma_0^-}) \leq 2(|\vec{v}_i| + k_{\Sigma_0^-})$. Since the type of w_i is determined uniquely by the types of a and elements of $\vec{M}_i, \vec{M}'_i$, finitely many possible forms of M exist.

By the construction, Σ'_0 is third-order, and $\mathrm{ord}(\mathscr{L} \circ \mathscr{L}') \leq \mathrm{ord}(\mathscr{L}) + \mathrm{ord}(\mathscr{L}') - 1 = \mathrm{ord}(\mathscr{L}) + 1$. Thus $\mathscr{G}^l \in \mathbf{G}(3, n+1)$ if $\mathscr{G} \in \mathbf{G}(3, n)$.

Definition 4.35. Let M be an abstract λ-term of an atomic type of semilexicalized third-order ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$. We say that a β-expansion $(\lambda x_1 \ldots x_m . M_0) M_1 \ldots M_m$ of a β-normal term $M \in \Lambda(\Sigma_0)$ is *good* if

- $M_0 \in \Lambda(\Sigma_0^+)$,
- every variable in M_0 other than x_i has an atomic type,

- $M_i \in \Gamma''(\Sigma_0^-)$ for $i \geq 1$,
- [*linking condition*] every path from a negative occurrence of an atomic type in the type of x_i ends in the type of an occurrence of a lexical constant in M_0.

If $M \in \mathcal{A}(\mathscr{G})$, then every variable in M has an atomic type, since $\mathscr{G}$ is third-order. It is clear that M has a good β-expansion.

Lemma 4.36. $\mathcal{O}(\mathscr{G}) = \mathcal{O}(\mathscr{G}^l)$.

Proof. The inclusion $\mathcal{O}(\mathscr{G}^l) \subseteq \mathcal{O}(\mathscr{G})$ is easily seen. If $P \in \mathcal{A}(\mathscr{G}^l)$, then $\mathscr{L}'(P) \in \mathcal{A}(\mathscr{G})$ and thus $\mathscr{L} \circ \mathscr{L}'(P) \in \mathcal{O}(\mathscr{G})$. We show the converse $\mathcal{O}(\mathscr{G}) \subseteq \mathcal{O}(\mathscr{G}^l)$. Recall that every $M \in \mathcal{A}(\mathscr{G})$ has a good β-expansion. By induction on M, we show that if a β-normal term $M \in \Lambda(\Sigma_0)$ of an atomic type has a good β-expansion, then we have $P \in \Lambda(\Sigma_0')$ such that $\tau_0'(P) = [\tau_0(M)]$, $\mathscr{L}'(P) = M$, and moreover, for each free variable y in M, P has the corresponding variable of type $[\tau_0(y)] \in \mathscr{A}_0'$.

Let $(\lambda\vec{x}.M_0)\vec{M}$ be a good β-expansion of M. M_0 has an atomic type. We have four cases depending on the head of M_0.

Case 1. The head of M_0 is a variable z not in $\vec{x}$. By the definition, every variable appearing in M_0 other than elements of $\vec{x}$ has an atomic type. That is, $M_0 = z^p$ for some $p \in \mathscr{A}_0$ and thus both $\vec{x}$ and $\vec{M}$ are empty. Let $P = z^{[p]}$.

Case 2. If the head of M_0 is a lexical constant $\mathsf{a} \in \mathscr{C}_0^+$ of type $\gamma_1 \to \cdots \to \gamma_n \to p$, then

$$\begin{aligned} M &= M_0[M_{i,j}/x_{i,j}]_{1\leq j\leq m_i}^{1\leq i\leq n} \\ &= \mathsf{a}(\lambda\vec{v}_1.N_{1,0})\ldots(\lambda\vec{v}_n.N_{n,0})[M_{i,j}/x_{i,j}]_{1\leq j\leq m_i}^{1\leq i\leq n} \\ &= \mathsf{a}(\lambda\vec{v}_1.N_{1,0}[M_{1,j}/x_{1,j}]_{1\leq j\leq m_1})\ldots(\lambda\vec{v}_n.N_{n,0}[M_{n,j}/x_{n,j}]_{1\leq j\leq m_n}) \\ &= \mathsf{a}N_1\ldots N_n \end{aligned} \tag{4.15}$$

where $\tau_0(N_{i,0}) \in \mathscr{A}_0$ and $(\lambda\vec{x}.M_0)\vec{M} = (\lambda\langle x_{i,j}\rangle_{1\leq j\leq m_i}^{1\leq i\leq n}.M_0)\langle M_{i,j}\rangle_{1\leq j\leq m_i}^{1\leq i\leq n}$. Suppose that $x_{i,j}$ appears in $N_{i,0}$ as

$$x_{i,j}\vec{L}_{i,j}$$

where $\tau_0(x_{i,j}\vec{L}_{i,j}) \in \mathscr{A}_0$.[7] Some elements of $\vec{L}_{i,j}$ are elements of $\vec{v}_i$ and some are not. Without loss of generality, we can assume that[8]

$$x_{i,j}\vec{L}_{i,j} = x_{i,j}\vec{v}_{i,j}\vec{L}'_{i,j}$$

[7] If $N_{i,0}$ is in long normal form, a subterm of that form is found. We can assume it without loss of generality.

[8] A permutation of the arguments of $x_{i,j}$ can be performed with the same permutation on the order of the abstraction in $M_{i,j}$.

where every element of $\vec{v}_{i,j}$ is in $\vec{v}_i$ but no element of $\vec{L}'_{i,j}$ is. Let $\vec{q}_{i,j} = \tau_0(\vec{v}_{i,j})$ and $\vec{q}'_{i,j} = \tau_0(\vec{L}'_{i,j})$. Thus, for

$$\tau_0(x_{i,j}) = \vec{q}_{i,j} \to \vec{q}'_{i,j} \to q_{i,j} = \tau_0(\vec{v}_{i,j}) \to \tau_0(\vec{L}'_{i,j}) \to q_{i,j},$$

every path starts from $\vec{q}_{i,j}$ ends in the type of the head occurrence of a in M_0, and no path starts from $\vec{q}'_{i,j}$ does. Corresponding to this partition, let

$$M_{i,j} = \lambda \vec{z}_{i,j} \vec{z}'_{i,j}.M^-_{i,j}$$

where $\tau_0(\vec{z}_{i,j}) = \vec{q}_{i,j}$, $\tau_0(\vec{z}'_{i,j}) = \vec{q}'_{i,j}$, $\tau_0(M^-_{i,j}) = q_{i,j}$.

Let us partition $\{1, \ldots, m_i\}$ into two subsets

$$I_i = \{\, j \mid 1 \leq j \leq m_i,\ \vec{q}'_{i,j} \neq \varepsilon \,\},$$
$$J_i = \{\, j \mid 1 \leq j \leq m_i,\ \vec{q}'_{i,j} = \varepsilon \,\}.$$

For each $j \in I_i$, by applying Lemma 4.33 to $M^-_{i,j}$ by regarding $\vec{z}_{i,j}$ as Z and $\vec{z}'_{i,j}$ as Y, we can find $k_{ij} \in \mathbb{N}$, $M''_{i,j}$ and $M_{i,j,h}$ for $1 \leq h \leq k_{ij}$ such that

- $M''_{i,j} \in \Gamma(\Sigma_0^-)$,
- $\vec{z}'_{i,j}$ consists of exactly the members of $\mathrm{Fv}^0(M''_{i,j})$,
- $M_{i,j,h} \in \Gamma'(\Sigma_0^-)$ for $1 \leq h \leq k_{ij}$,
- $\vec{z}_{i,j} = \vec{z}_{i,j,1} \ldots \vec{z}_{i,j,k_{ij}}$ where $\vec{z}_{i,j,h}$ consists of exactly the members of $\mathrm{Fv}^0(M_{i,j,h})$,
- $M^-_{i,j} = M''_{i,j}[M_{i,j,h}/y_{i,j,h}]_{1 \leq h \leq k_{ij}}$.

We let

$$M'_{i,j} = \lambda \vec{z}'_{i,j}.M''_{i,j} \in \Gamma''(\Sigma_0^-),$$

so

$$\begin{aligned} M_{i,j} &= \lambda \vec{z}_{i,j} \vec{z}'_{i,j}.M^-_{i,j} \\ &= \lambda \vec{z}_{i,j} \vec{z}'_{i,j}.M''_{i,j}[M_{i,j,h}/y_{i,j,h}]_{1 \leq h \leq k_{ij}} \\ &= \lambda \vec{z}_{i,j}.M'_{i,j}[M_{i,j,h}/y_{i,j,h}]_{1 \leq h \leq k_{ij}} \end{aligned} \tag{4.16}$$

for $j \in I_i$.

Let $N'_{i,0}$ be obtained from $N_{i,0}$ by replacing $x_{i,j}\vec{v}_{i,j}$ with $x'_{i,j}$, i.e.,

$$N_{i,0} = N'_{i,0}[x_{i,j}\vec{v}_{i,j}/x'_{i,j}]_{1 \leq j \leq m_i}, \tag{4.17}$$

where $\tau_0(x'_{i,j}) = \vec{q}'_{i,j} \to q_{i,j}$. If $j \in J_i$, then $\tau_0(x'_{i,j}) = q_{i,j} \in \mathscr{A}_0$. Let

$$N'_i = N'_{i,0}[M'_{i,j}/x'_{i,j}]_{j \in I_i}. \tag{4.18}$$

We show that $(\lambda\langle x'_{i,j}\rangle_{j\in I_i}.N'_{i,0})\langle M'_{i,j}\rangle_{j\in I_i}$ is a good β-expansion of N'_i. By $M_0 \in \Lambda(\Sigma_0^+)$, $N_{i,0} \in \Lambda(\Sigma_0^+)$, thus $N'_{i,0} \in \Lambda(\Sigma_0^+)$, and $\tau_0(N'_{i,0}) = \tau_0(N_{i,0}) \in \mathscr{A}_0$. All variables appearing in $N_{i,0}$ other than $x'_{i,j}$ appears in $M_0, N_{i,0}$, so they have atomic types, and $x'_{i,j}$ with $j \in J_i$ also have atomic types. $M'_{i,j} \in \Gamma''(\Sigma_0^-)$ by its construction. Recall that for $\tau_0(x_{i,j}) = \vec{q}_{i,j} \to \vec{q}'_{i,j} \to q_{i,j}$, all occurrences of atomic types in $\vec{q}_{i,j}$ are linked to the head occurrence of a in M_0, and all occurrences of atomic types in $\vec{q}'_{i,j}$ are linked to some other lexical constants in M_0, i.e., those lexical constants are in $N_{i,0}$ and thus in $N'_{i,0}$. This implies that, for $\tau_0(x'_{i,j}) = \vec{q}'_{i,j} \to q_{i,j}$, all occurrences of atomic types in $\vec{q}'_{i,j}$ are linked to those lexical constants in $N'_{i,0}$. Therefore, $(\lambda\langle x'_{i,j}\rangle_{j\in I_i}.N'_{i,0})\langle M'_{i,j}\rangle_{j\in I_i}$ satisfies the linking condition and thus is a good β-expansion of N'_i. By applying the induction hypothesis to N'_i, we get $P'_i \in \Lambda(\Sigma'_0)$ such that $\tau'_0(P'_i) = [\tau_0(N'_i)]$,

$$\mathscr{L}'(P'_i) = N'_i, \tag{4.19}$$

and each free variable y of N'_i has the type $[\tau_0(y)] \in \mathscr{A}'_0$ in P'_i. Let

- $\vec{v}'_i$ be the subsequence of $\vec{v}_i$ consisting of variables not in $\vec{v}_{i,j}$ for any j,
- $\vec{x}'_i$ be the sequence of variables $x'_{i,j}$ for $j \in J_i$,
- $\vec{y}_i$ be the sequence of variables $y_{i,j,h}$ for $j \in I_i$, $1 \le h \le k_{ij}$.

Indeed the variables constituting above sequences occur free in N'_i. Let

$$P_i = \lambda\vec{v}'_i\vec{x}'_i\vec{y}_i.P'_i, \tag{4.20}$$

where

$$\tau'_0(P_i) = [\tau_0(\vec{v}'_i)] \to [\tau_0(\vec{x}'_i)] \to [\tau_0(\vec{y}_i)] \to [\tau_0(N'_i)].$$

On the other hand, the following equation holds:

$$\begin{aligned}
N_i &= \lambda\vec{v}_i.N_{i,0}[M_{i,j}/x_{i,j}]_{1\le j\le m_i} && \text{(by (4.15))}\\
&= \lambda\vec{v}_i.N'_{i,0}[x_{i,j}\vec{v}_{i,j}/x'_{i,j}]_{1\le j\le m_i}[M_{i,j}/x_{i,j}]_{1\le j\le m_i} && \text{(by (4.17))}\\
&= \lambda\vec{v}_i.N'_{i,0}[x_{i,j}\vec{v}_{i,j}/x'_{i,j}]_{1\le j\le m_i}[\lambda\vec{z}_{i,j}.M'_{i,j}[M_{i,j,h}/y_{i,j,h}]_{1\le h\le k_{ij}}/x_{i,j}]_{j\in I_i}\\
&\qquad\qquad [M_{i,j}/x_{i,j}]_{j\in J_i} && \text{(by (4.16))}\\
&= \lambda\vec{v}_i.N'_{i,0}[M'_{i,j}[M_{i,j,h}[\vec{v}_{i,j,h}/\vec{z}_{i,j,h}]/y_{i,j,h}]_{1\le h\le k_{ij}}/x'_{i,j}]_{j\in I_i}[M_{i,j}\vec{v}_{i,j}/x'_{i,j}]_{j\in J_i}\\
&= \lambda\vec{v}_i.N'_i[M_{i,j,h}[\vec{v}_{i,j,h}/\vec{z}_{i,j,h}]/y_{i,j,h}]_{j\in I_i}^{1\le h\le k_{ij}}[M_{i,j}\vec{v}_{i,j}/x'_{i,j}]_{j\in J_i} && \text{(by (4.18))}
\end{aligned}$$

where $\vec{v}_{i,j,h}$ is the subsequence of $\vec{v}_{i,j}$ corresponding to the subsequence $\vec{z}_{i,j,h}$ of $\vec{z}_{i,j}$. Letting

$$\vec{M}_i = \langle |M_{i,j}\vec{v}_{i,j}|_\beta \rangle_{j \in J_i},$$
$$\vec{M}'_i = \langle M_{i,j,h}[\vec{v}_{i,j,h}/\vec{z}_{i,j,h}] \rangle_{j \in I_i}^{1 \leq h \leq k_{ij}},$$

we have

$$N_i = \lambda \vec{v}_i . N'_i[\vec{M}_i/\vec{x}'_i][\vec{M}'_i/\vec{y}_i]. \tag{4.21}$$

Note that $|M_{i,j}\vec{v}_{i,j}|_\beta \in \Gamma(\Sigma_0^-)$ for $j \in J_i$, and $M_{i,j,h}[\vec{v}_{i,j,h}/\vec{z}_{i,j,h}] \in \Gamma'(\Sigma_0^-)$ for $j \in I_i$. Let

$$L = \lambda \vec{y} w_1 \dots w_n . \mathsf{a}(\lambda \vec{v}_1 . w_1 \vec{v}'_1 \vec{M}_1 \vec{M}'_1) \dots (\lambda \vec{v}_n . w_n \vec{v}'_n \vec{M}_n \vec{M}'_n)$$

where $\vec{y}$ consists of free variables of $\vec{M}_i$ and $\vec{M}'_i$ other than elements of $\vec{v}_i$. Since $K = \lambda z.z \in \Gamma''_1(\Sigma_0)$, by the definition, we have a constant $\mathsf{A}_{K,L} \in \mathscr{C}'_0$ such that

$$\tau'_0(\mathsf{A}_{K,L}) = [\tau_0(\vec{y})] \to \tau'_0(P_1) \to \dots \to \tau'_0(P_n) \to [p]$$
$$\mathscr{L}'(\mathsf{A}_{K,L}) = L$$

Thus, for $P = \mathsf{A}_{K,L}\vec{y}P_1 \dots P_n \in \Lambda(\Sigma'_0)$, we have

$$\begin{aligned}
\mathscr{L}'(P) &= \mathscr{L}'(\mathsf{A}_{K,L}\vec{y}P_1 \dots P_n) \\
&= \mathscr{L}'(\mathsf{A}_{K,L}\vec{y})(\lambda \vec{v}'_1 \vec{x}'_1 \vec{y}_1 . N'_1) \dots (\lambda \vec{v}'_n \vec{x}'_n \vec{y}_n . N'_n) \qquad \text{(by (4.19),(4.20))} \\
&= \mathsf{a}(\lambda \vec{v}_1 . N'_1[\vec{M}_1/\vec{x}'_1][\vec{M}'_1/\vec{y}_1]) \dots (\lambda \vec{v}_n . N'_n[\vec{M}_n/\vec{x}'_n][\vec{M}'_n/\vec{y}_n]) \\
&= \mathsf{a}N_1 \dots N_n \qquad \text{(by (4.21))} \\
&= M.
\end{aligned}$$

Case 3. Suppose that $M_0 = x_1 M'_0$ for a unary variable x_1 in $\vec{x}$. Then the head of M'_0 is a lexical constant a by the linking condition. Let M_1 be the corresponding term in $\vec{M}$ to x_1. Since x_1 and M_1 are unary, either $M_1 = \lambda z.z$ or $M_1 = \lambda z.y_0 z$ for some unary variable y_0. Let $\vec{M}'$ be $\vec{M}$ minus M_1 and $\vec{x}'$ be $\vec{x}$ minus x_1.

If $M_1 = \lambda z.z$, then this implies that M has a good β-expansion $(\lambda \vec{x}'.M'_0)\vec{M}'$. This case reduces to Case 2.

Suppose that $M_1 = \lambda z.y_0 z$. M can be represented as

$$M = x_1 M'_0[M_1/x_1, \vec{M}'/\vec{x}'] = y_0 M'_0[\vec{M}'/\vec{x}'] = y_0(\mathsf{a}N_1 \dots N_n). \tag{4.22}$$

Applying exactly the same discussion as Case 2 to $M'_0[\vec{M}'/\vec{x}']$, for each $i \in \{1, \dots, n\}$, we get a λ-term N'_i and sequences of λ-terms $\vec{M}_i$ and $\vec{M}'_i$ such that

N'_i has a good β-expansion, each element of $\vec{M}_i$ is in $\Gamma(\Sigma_0^-)$, each element of $\vec{M}'_i$ is in $\Gamma'(\Sigma_0^-)$, and

$$N_i = \lambda\vec{v}_i.N'_i[\vec{M}_i/\vec{x}'_i][\vec{M}'_i/\vec{y}_i], \tag{4.23}$$

and moreover, we have $P_i \in \Lambda(\Sigma'_0)$ such that

$$\mathscr{L}'(P_i) = \lambda\vec{v}'_i\vec{x}'_i\vec{y}_i.N'_i \tag{4.24}$$

where $\vec{v}'_i$ consists of variables in $\vec{v}_i$ which appear in neither $\vec{M}_i$ nor $\vec{M}'_i$, and every free variable y in N'_i has type $[\tau_0(y)] \in \mathscr{A}'_0$ in P_i. Since $M_1 = \lambda z.y_0 z \in \Gamma''_1(\Sigma_0^-)$, by the definition, we have a constant B such that

$$\begin{aligned}\tau'_0(\mathsf{B}) &= [\tau_0(\vec{y})] \to [\tau_0(y_0)] \to \tau'_0(P_1) \to \cdots \to \tau'_0(P_n) \to q\\ \mathscr{L}'(\mathsf{B}) &= \lambda\vec{y}y_0w_1\dots w_n.y_0(\mathsf{a}(\lambda\vec{v}_1.w_1\vec{v}'_1\vec{M}_1\vec{M}'_1)\dots(\lambda\vec{v}_n.w_n\vec{v}'_n\vec{M}_n\vec{M}'_n))\end{aligned} \tag{4.25}$$

where $\vec{y}$ consists of free variables of $\vec{M}_i$ or $\vec{M}'_i$ other than elements of $\vec{v}_i$. Thus, for

$$P = \mathsf{B}\vec{y}y_0P_1\dots P_n,$$

we have

$$\begin{aligned}\mathscr{L}'(P) &= \mathscr{L}'(\mathsf{B})\vec{y}y_0(\lambda\vec{v}'_1\vec{x}'_1\vec{y}_1.N'_1)\dots(\lambda\vec{v}'_n\vec{x}'_n\vec{y}_n.N'_n) && \text{(by (4.24))}\\ &= y_0(\mathsf{a}(\lambda\vec{v}_1.N'_1[\vec{M}_1/\vec{x}'_1][\vec{M}'_1/\vec{y}_1])\dots(\lambda\vec{v}_n.N'_n[\vec{M}_n/\vec{x}'_n][\vec{M}'_n/\vec{y}_n])) && \text{(by (4.25))}\\ &= y_0(\mathsf{a}N_1\dots N_n) && \text{(by (4.23))}\\ &= M. && \text{(by (4.22))}\end{aligned}$$

Case 4. Suppose that the head of M_0 is a variable x_1 in $\vec{x}$ and x_1 is a k-ary variable for some $k \geq 2$. Let the corresponding term M_1 in $\vec{M}$ be $M_1 = \lambda z_1\vec{z}.M''_1$ with $|z_1\vec{z}| = k$ and $\tau_0(M''_1) \in \mathscr{A}_0$. By applying Lemma 4.33 to M''_1 with respect to $Y = \mathrm{Fv}^0(M''_1) - \{z_1\} \neq \varnothing$ and $Z = \{z_1\}$, we can find $M'''_1 \in \Gamma(\Sigma_0^-)$ and $K \in \Gamma'(\Sigma_0^-)$ such that $M''_1 = M'''_1[K/y_0]$. By $|\mathrm{Fv}^0(K)| = 1$, $\lambda z_1.K \in \Gamma''_1(\Sigma_0^-)$. Let $M'_1 = \lambda\vec{z}.M'''_1 \in \Gamma''(\Sigma_0^-)$. We have

$$M_1 = \lambda z_1.M'_1[K/y_0]. \tag{4.26}$$

Let $M_0 = x_1L_1\dots L_k$ (recall that M_0 has an atomic type) and $\vec{x}$ divide into three subsequences x_1, $\vec{x}'$, $\vec{x}''$ so that $\vec{x}'$ consists of variables in L_1, $\vec{x}''$ consists of variables in L_j for some $j \geq 2$. Corresponding to this division, $\vec{M}$

is divided into three subsequences M_1, $\vec{M'}$, and $\vec{M''}$. M can be represented as

$$\begin{aligned} M &= M_0[\vec{M}/\vec{x}] \\ &= x_1 L_1 L_2 \dots L_k[M_1/x_1, \vec{M'}/\vec{x'}, \vec{M''}/\vec{x''}] \\ &= M_1 L_1 L_2 \dots L_k[\vec{M'}/\vec{x'}][\vec{M''}/\vec{x''}] \\ &= M'_1[K[L_1[\vec{M'}/\vec{x'}]/z_1]/y_0] L_2 \dots L_k[\vec{M''}/\vec{x''}] \end{aligned} \tag{4.27}$$

by (4.26). Let

$$\begin{aligned} M' &= M'_1 L_2 \dots L_k[\vec{M''}/\vec{x''}], \\ M'_0 &= x'_1 L_2 \dots L_k \quad (\text{thus } M_0 = M'_0[x_1 L_1/x'_1]) \end{aligned} \tag{4.28}$$

where $\tau_0(x'_1) = \tau_0(M'_1)$. By (4.27) and (4.28),

$$M = M'[K[L_1[\vec{M'}/\vec{x'}]/z_1]/y_0]. \tag{4.29}$$

By $K \in \Gamma'(\Sigma_0^-)$, it contains at least one occurrence of a lexical constant or a variable of l-ary with $l \geq 2$. Therefore, the size of M' is strictly smaller than M. It is not difficult to see that M' has a good β-expansion $(\lambda x'_1 \vec{x''}.M'_0)M'_1\vec{M''}$, since $(\lambda\vec{x}.M_0)\vec{M}$ is a good β-expansion of M. By the induction hypothesis, we have $P' \in \Lambda(\Sigma'_0)$ such that $\tau'_0(P') = [\tau_0(M')]$,

$$\mathscr{L}'(P') = M', \tag{4.30}$$

and $\tau'_0(y) = [\tau_0(y)] \in \mathscr{A}'_0$ for every free variable y in M'. In particular, $\tau'_0(y_0) = [\tau_0(y_0)]$.

By the linking condition on x_1, the head of L_j is a lexical constant for each $j \in \{1, \dots, k\}$. Particularly for $j = 1$, there are $N_{1,0}, \dots, N_{n,0}$ such that L_1 can be written as

$$L_1 = \mathsf{a}(\lambda\vec{v}_1.N_{1,0}) \dots (\lambda\vec{v}_n.N_{n,0}).$$

Letting $N_1, \dots, N_n$ be such that

$$L_1[\vec{M'}/\vec{x'}] = \mathsf{a}N_1 \dots N_n, \tag{4.31}$$

we have

$$\begin{aligned} M &= M'[K[L_1[\vec{M'}/\vec{x'}]/z_1]/y_0] \\ &= M'[K[\mathsf{a}N_1 \dots N_n/z_1]/y_0] \end{aligned} \tag{4.32}$$

by (4.29) and (4.31). It is not difficult to see that $(\lambda\vec{x}'.L_1)\vec{M}'$ is a good β-expansion of $L_1[\vec{M}'/\vec{x}'] = \mathsf{a}N_1 \dots N_n$, since $(\lambda\vec{x}.M_0)\vec{M}$ is a good β-expansion of M. Applying exactly the same discussion as Case 2 to $L_1[\vec{M}'/\vec{x}']$, for each $i \in \{1, \dots, n\}$, we have a λ-term N_i' and sequences $\vec{M}_i$ and $\vec{M}_i'$ of λ-terms such that N_i' has a good β-expansion, each element of $\vec{M}_i$ is in $\Gamma(\Sigma_0^-)$, each element of $\vec{M}_i'$ is in $\Gamma'(\Sigma_0^-)$, and

$$N_i = \lambda\vec{v}_i.N_i'[\vec{M}_i/\vec{x}_i'][\vec{M}_i'/\vec{y}_i], \tag{4.33}$$

and moreover, we have $P_i \in \Lambda(\Sigma_0')$ such that

$$\mathscr{L}'(P_i) = \lambda\vec{v}_i'\vec{x}_i'\vec{y}_i.N_i' \tag{4.34}$$

where $\vec{v}_i'$ consists of variables in $\vec{v}_i$ which appear in neither $\vec{M}_i$ nor $\vec{M}_i'$, and every free variable y in N_i' has type $[\tau_0(y)]$ in P_i. By the definition, we have a constant C such that

$$\begin{aligned}
&\tau_0'(\mathsf{C}) = [\tau_0(\vec{y})] \to \tau_0'(P_1) \to \dots \to \tau_0'(P_n) \to [\tau_0(K)] \\
&\mathscr{L}'(\mathsf{C}) = \lambda\vec{y}w_1 \dots w_n.K[\mathsf{a}(\lambda\vec{v}_1.w_1\vec{v}_1'\vec{M}_1\vec{M}_1') \dots (\lambda\vec{v}_n.w_n\vec{v}_n'\vec{M}_n\vec{M}_n')/z_1]
\end{aligned} \tag{4.35}$$

where $\vec{y}$ consists of free variables of K, $\vec{M}_i$, or $\vec{M}_i'$ other than z_1, $\vec{v}_i$. Thus, for

$$P = P'[\mathsf{C}\vec{y}P_1 \dots P_n/y_0],$$

we have

$$\begin{aligned}
\mathscr{L}'(P) &= M'[\mathscr{L}'(\mathsf{C})\vec{y}(\lambda\vec{v}_1'\vec{x}_1'\vec{y}_1.N_1') \dots (\lambda\vec{v}_n'\vec{x}_n'\vec{y}_n.N_n')/y_0] && \text{(by (4.30), (4.34))} \\
&= M'[K[\mathsf{a}(\lambda\vec{v}_1.N_1'[\vec{M}_1/\vec{x}_1'][\vec{M}_1'/\vec{y}_1]) \\
&\qquad \dots (\lambda\vec{v}_n.N_n'[\vec{M}_n/\vec{x}_n'][\vec{M}_n'/\vec{y}_n])/z_1]/y_0] && \text{(by (4.35))} \\
&= M'[K[\mathsf{a}N_1 \dots N_n/z_1]/y_0] && \text{(by (4.33))} \\
&= M. && \text{(by (4.32))}
\end{aligned}$$

□

Theorem 4.37. *For every semilexicalized ACG $\mathscr{G} \in \mathbf{G}(3, n)$, there is a lexicalized ACG $\mathscr{G}' \in \mathbf{G}(3, n+1)$ such that*

$$\mathcal{O}(\mathscr{G}') = \{\, R \in \mathcal{O}(\mathscr{G}) \mid R \text{ contains a constant} \,\}.$$

4.3.6 Schabes et al.'s Lexicalization and Ours

Although our notion of lexicalization differs from the one by Schabes et al. [47, 48], what we have done is not very far from their definition. They define the notion of lexicalization in a strong sense, where it preserves not only the string language but also the tree language yielding the string language. This definition requires grammars that should be lexicalized to be finitely ambiguous, i.e., the tree language contains at most finitely many trees that yield the same string. Their motivation behind the definition would be to preserve the syntactic structures as well as the surface structures. If we regard both tree languages and abstract languages as syntactic structures that yield string languages and object languages as surface structures, respectively, one might think the definition of "lexicalization of ACGs" should be strengthened in analogy with Schabes et al.'s definition so that the abstract languages of the given ACG and the resultant lexicalized ACG coincide. However, this definition seems too rigorous and there would be very few interesting subclasses of ACGs admit strong lexicalization in this sense.

We alternatively propose to define the notion of "strong lexicalization of ACGs" analogous to Schabes et al.'s definition so that an ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$ is converted into a lexicalized ACG $\mathscr{G}' = \langle \Sigma'_0, \Sigma_1, \mathscr{L} \circ \mathscr{L}', s' \rangle$ where $\mathcal{A}(\mathscr{G}) = \mathscr{L}'(\mathcal{A}(\mathscr{G}'))$ holds. According to this definition, one can "recover" the original abstract language from the resultant ACG. Actually, this is what we have done in the latter half of our lexicalization method on the preprocessed ACGs which have neither nullary nor unary nonlexical constants. Moreover, when we assume that input semilexicalized ACGs are finitely ambiguous in the sense that every element of the object language has finitely many abstract terms mapped to it, it is possible to modify the preprocessing that eliminates nullary and unary nonlexical constants so that the resultant grammar can also recover the original abstract language. Conversely, one may think of the preprocessing as elimination of infinite ambiguity from the given semilexicalized ACG. This way our whole algorithm becomes a strong lexicalization method. Therefore, our lexicalization method has a close conceptual similarity to Schabes et al.'s notion of lexicalization.

In the sequel, we concretely see the relation between Schabes et al.'s lexicalization and ours through lexicalization of finitely ambiguous CFGs. Here our lexicalization method means modified lexicalization method, which recovers the original abstract languages. As Schabes et al.'s lexicalization method converts a finitely ambiguous CFG into a "strongly equivalent" lexicalized TAG in the sense that the tree language of the TAG is the set of derivation trees of the CFG, our lexicalization method for semilexicalized ACGs converts the ACG $\mathscr{G}_G \in \mathbf{G}_{\text{string}}(2,2)$ encoding a finitely ambiguous CFG

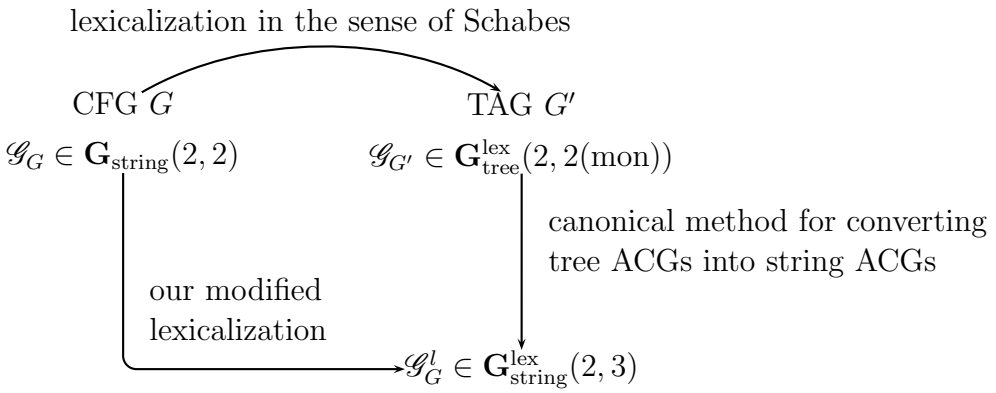


Figure 4.1: Our lexicalization of a CFG

G into the ACG $\mathscr{G}^l_G \in \mathbf{G}^{\text{lex}}_{\text{string}}(2,3)$ encoding a lexicalized TAG G' strongly equivalent to G "as a string generator". That a string ACG $\mathscr{G}^l_G$ encodes a TAG G' as a string generator means that $\mathscr{G}^l_G$ is obtained from the ACG $\mathscr{G}_{G'} \in \mathbf{G}^{\text{lex}}_{\text{tree}}(2, 2(\text{mon}))$ encoding G' by the canonical method for transforming a tree ACG into a string ACG generating the yield string language of the original tree ACG (see Proposition 5.5 for the canonical method). Since such $\mathscr{G}_{G'}$ and G' are effectively computable from $\mathscr{G}^l_G$, our method gives alternative construction for Schabes's theorem that every finitely ambiguous CFG has a strongly equivalent lexicalized TAG. This is illustrated in Figure 4.1.

More direct comparison would be possible when modifying the definition of "lexicalized ACGs". For the accordance with Schabes et al.'s notion of lexicalized grammar, here we introduce the notion of "strongly lexicalized tree ACGs". We say that an abstract constant of a tree ACG is *strongly lexical* iff it is mapped to a term containing an object constant of an atomic type. Then the set $\mathscr{C}_0$ of abstract constants is partitioned into $\mathscr{D}_0$ and $\mathscr{E}_0$ where $\mathscr{D}_0$ is the set of strongly lexical abstract constants. Note that $\mathscr{C}_0^- \subseteq \mathscr{E}_0$ and $\mathscr{D}_0 \subseteq \mathscr{C}_0^+$. We then call the tree ACG *strongly lexicalized* iff $\mathscr{E}_0$ is the empty set. It is not difficult to modify the first half of our lexicalization method, which eliminates nullary and unary constants in $\mathscr{C}_0^-$, so that it eliminates nullary and unary constants in $\mathscr{E}_0$, if the input ACG encodes a finitely ambiguous CFG in the sense of Schabes et al. Besides, recall that the second half of our lexicalization method depends only on the fact that every constant in $\mathscr{C}_0^-$ has a k-ary second-order type for some $k \geq 2$. Thus, replacing $\mathscr{C}_0^+$ with $\mathscr{D}_0$ and $\mathscr{C}_0^-$ with $\mathscr{E}_0$ in our procedure, we get a strong lexicalization method that completely satisfies Schabes et al.'s notion of lexicalization of CFGs.

4.4 Summary

This chapter has discussed lexicalized ACGs, in particular, lexicalization of semilexicalized ACGs. As seen in Section 4.2, to be lexicalized can be thought of as a desirable restriction on ACGs not only from the lexicalists' point of view, but also from the point of view of the computational complexity.

In section 4.3, we have presented a lexicalization method for semilexicalized ACGs, where every nonlexical abstract constant has a type of at most second-order. Our basic strategy for lexicalizing semilexicalized ACGs is as follows. As a preparation, we first eliminate nullary or unary nonlexical constants from the given grammar. Then for the resultant grammar $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$, we construct a new ACG $\mathscr{G}' = \langle \Sigma_0', \Sigma_0, \mathscr{L}', s' \rangle$ so that $\mathcal{O}(\mathscr{G}') = \mathcal{A}(\mathscr{G})$ and $\mathscr{L}'(\mathsf{A})$ contains a lexical abstract constant in $\mathscr{C}_0^+$ with some finite number of nonlexical constants in $\mathscr{C}_0^-$ for every $\mathsf{A} \in \mathscr{C}_0'$. We have presented three kinds of lexicalization methods depending on the order of given ACGs. While the lexicalization method for ACGs with fourth or higher-order abstract vocabularies preserves the orders of the lexicons, our lexicalization methods for ACGs with second and third-order abstract vocabularies both increase the orders of the lexicons by one.

Our result gives another proof for the theorem that every semilexicalized ACG generates a language in NP, which has been proven by Salvati [43].

Moreover, our lexicalization method for second-order ACGs, though it does not preserves the orders of the lexicons, entails that every LCFRS has an equivalent lexicalized LCFRS modulo the empty string, where an LCFRS is said to be *lexicalized* iff the description of each function assigned to each production includes at least one terminal symbol. It is not difficult to modify Salvati's conversion [45] from a string second-order ACG $\mathscr{G} \in \mathbf{G}_{\text{string}}(2, n)$ into an equivalent LCFRS so that it preserves lexicalization. Therefore, the ACG $\mathscr{G} \in \mathbf{G}_{\text{string}}(2, 4)$ encoding an LCFRS, which has an equivalent lexicalized ACG $\mathscr{G}' \in \mathbf{G}^{\text{lex}}_{\text{string}}(2, 5)$ modulo the empty string by our lexicalization, has an equivalent lexicalized LCFRS modulo the empty string.

For the sake of generality, we have defined the notion of lexicalized ACGs so that it does not comprehend Schabes et al.'s definition of lexicalized TAGs [47, 48]. Nevertheless, as discussed in Section 4.3.6, our lexicalization method can be modified so that it entails Schabes et al.'s theorem that every finitely ambiguous CFG has a strongly equivalent lexicalized TAG.

Therefore, our lexicalization method partially generalizes the research by Schabes et al. The result of this chapter, as well as the previous chapter, strengthens de Groote and Pogodalla's view that ACGs can be the kernel of a grammatical framework; not only well known grammar formalisms themselves, but also transformations of existing grammar formalisms are encoded

in ACGs.

Future Work

It is future work whether or not we can lexicalize second and/or third-order ACGs with preserving the orders of the lexicons. It is known that several subclasses of second-order string ACGs are closed under lexicalization, where the order of the lexicons is preserved.

Recall that $\mathbf{G}_{\text{string}}(2,2)$ is equivalent to the class of context-free grammars. According to the straightforward encoding by de Groote and Pogodalla [15, 19], if a CFG is "lexicalized" in the sense that every production contains a terminal symbol in its right-hand side, then the corresponding second-order ACG is lexicalized, and vice versa. The fact that every CFG that does not generate the empty string has an equivalent lexicalized CFG, e.g., Greibach normal form, implies that every $\mathscr{G} \in \mathbf{G}_{\text{string}}(2,2)$ has an equivalent $\mathscr{G}' \in \mathbf{G}^{\text{lex}}_{\text{string}}(2,2)$ modulo the empty string.

As mentioned above, every string second-order ACG $\mathscr{G} \in \mathbf{G}_{\text{string}}(2,n)$ has an equivalent lexicalized LCFRS modulo the empty string, which has an equivalent lexicalized ACG $\mathscr{G}' \in \mathbf{G}^{\text{lex}}_{\text{string}}(2,4)$.

Moreover, by Schabes et al.'s work on lexicalization of TAGs [47,48], it is not hard to see that the subclass of second-order string ACGs whose lexicons map an atomic type to either str or $\mathit{str} \to \mathit{str}$ is closed under lexicalization.

The reason why we cannot preserve the order of the lexicon is that our lexicalization method is designed to be able to "recover" the original abstract language, in the sense that there is a lexicon from the new abstract vocabulary to the original abstract vocabulary such that the original abstract language is exactly the image of the new abstract language, except for the steps eliminating nullary and unary nonlexical constants. In order to lexicalize second-order ACGs with preserving the orders of the lexicons, it seems inevitable to give up recovering the original abstract languages.

If each subclass $\mathbf{G}(2,n)$ is closed under lexicalization, it might entail that some grammar formalisms, e.g., linear CFTGs, admit lexicalization, with or without help of a similar discussion in Section 4.3.6.

On the other hand, it seems difficult to lexicalize general third-order ACGs, because there is a third-order ACG that simulates vector addition systems (Proposition 2.18. See also Proposition 5.7), while every lexicalized ACG generates an NPTIME language (Proposition 4.3).[9]

[9]The author could not find a reference on the complexity of a *fixed* Petri-net reachability set.

Chapter 5

Two-Dimensional Abstract Categorial Grammars

5.1 Introduction

If two ACGs $\mathscr{G}_1$ and $\mathscr{G}_2$ share the same abstract vocabulary and the distinguished type, i.e., $\mathcal{A}(\mathscr{G}_1) = \mathcal{A}(\mathscr{G}_2)$, each element P_1 of $\mathcal{O}(\mathscr{G}_1)$ is associated with terms P_2 in $\mathcal{O}(\mathscr{G}_2)$ through the elements M of $\mathcal{A}(\mathscr{G}_1) = \mathcal{A}(\mathscr{G}_2)$ which are mapped to P_1 and P_2. This way, ACGs can represent *relations* of two languages. It is convenient to give a formalization to this paradigm.

Definition 5.1. A *two-dimensional ACG (2D-ACG)* is a sextuple $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \Sigma_2, \mathscr{L}_1, \mathscr{L}_2, s \rangle$ such that

- Σ_0 is a higher-order signature, called the *abstract vocabulary*,
- Σ_i for $i \in \{1, 2\}$ is a higher-order signature, called the *i-th object vocabulary*,
- $\mathscr{L}_i$ is a lexicon from Σ_0 to Σ_i (i-th lexicon),
- $s \in \mathscr{A}_0$ is the distinguished type.

The *i-th projection* of $\mathscr{G}$ is defined as $\mathscr{G}_i = \langle \Sigma_0, \Sigma_i, \mathscr{L}_i, s \rangle$. The *abstract language*, *object language* and *i-th projection of the object language* are respectively defined as

$$\begin{aligned}
\mathcal{A}(\mathscr{G}) &= \{\, |M|_{\beta\eta} \in \Lambda(\Sigma_0) \mid M \text{ is a closed linear } \lambda\text{-term of type } s \,\} \\
\mathcal{O}(\mathscr{G}) &= \{\, \langle |\mathscr{L}_1(M)|_{\beta\eta}, |\mathscr{L}_2(M)|_{\beta\eta} \rangle \mid M \in \mathcal{A}(\mathscr{G}) \,\} \\
\mathcal{O}_i(\mathscr{G}) &= \{\, P_i \mid P_i \in \mathcal{O}(\mathscr{G}_i) \text{ for } \mathscr{G}_i \text{ the } i\text{-th projection of } \mathscr{G} \,\}
\end{aligned}$$

Moreover, the classification of (one-dimensional) ACGs can be used for 2D-ACGs. For instance, $\mathbf{G}_{\text{tree,string}}(2, 1(\text{r}), 4)$ represents the class of ACGs whose first projection belongs to $\mathbf{G}_{\text{tree}}(2, 1(\text{r}))$ and second projection belongs to $\mathbf{G}_{\text{string}}(2, 4)$.

This way we can define *k-dimensional ACGs (kD-ACGs)* for any k.

Though symmetric treatment of syntax and semantics of natural language is an important appeal of ACGs, there is little mathematical research on 2D-ACGs. In this chapter, we discuss the expressive power on *relations* of 2D-ACG. Although it is rather important what kind of relations between strings and meaning representations can be expressed by 2D-ACGs, we have no well-known measure to characterize the class of relations between strings and meanings. Therefore, we evaluate the expressive power of 2D-ACGs through encoding by 2D-ACGs of some well-known tree transducer formalisms that define relations between trees and trees, or trees and strings.

Section 5.2 presents some examples of 2D-ACGs, which imply some closure properties of string ACGs. A fundamental question is whether several mathematical properties of 1D-ACGs still hold for 2D-ACGs. We show that every second-order 2D-ACG generates a PTIME language as every second-order 1D-ACG does (Theorem 2.11) in Section 5.3. Moreover, we show that every second-order string 2D-ACG in $\mathbf{G}_{\text{string}}(2, n_1, n_2)$ has an equivalent 2D-ACG $\mathbf{G}_{\text{string}}(2, 4, 4)$, like second-order string 1D-ACGs (Theorem 2.6). This is shown in Section 5.5, after we prove that every *deterministic tree walking transducer* can be encoded by a second-order 2D-ACG. Section 5.4 shows that the class of *linear macro tree transducers* exactly corresponds to $\mathbf{G}_{\text{tree,string}}(2, 1(\text{sr}), 2)$.

Related Work

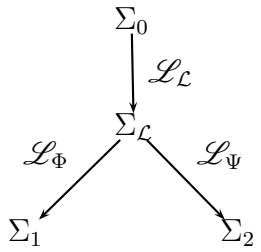
For a binary relation $\mathcal{R}$, *the domain* $\mathcal{R}_1$ *of* $\mathcal{R}$ is defined as

$$\mathcal{R}_1 = \{\, P \mid \langle P, Q\rangle \in \mathcal{R} \text{ for some } Q \,\}$$

and *the range* $\mathcal{R}_2$ *of* $\mathcal{R}$ is defined as

$$\mathcal{R}_2 = \{\, Q \mid \langle P, Q\rangle \in \mathcal{R} \text{ for some } P \,\}.$$

Finite state transducers are simple extensions of finite state-automata which write some possibly empty string for each transition. The output string by a transducer for an input string is defined as the concatenation of the strings written during the transition steps. A finite state transducer translates a regular language into a regular language. For a formal definition and basic properties of finite state transducers, see a standard text book (e.g., [3]). De Groote establishes the following result.



A bimorphism $\langle \mathcal{L}, \Phi, \Psi \rangle$ is represented by $\langle \Sigma_0, \Sigma_1, \Sigma_2, \mathscr{L}_\Phi \circ \mathscr{L}_{\mathcal{L}}, \mathscr{L}_\Psi \circ \mathscr{L}_{\mathcal{L}}, s \rangle$, if $\mathcal{L}$ is represented by $\langle \Sigma_0, \Sigma_{\mathcal{L}}, \mathscr{L}_{\mathcal{L}}, s \rangle$.

Figure 5.1: Bimorphism and ACG

Proposition 5.2 (de Groote [15]). *For every finite state transducer T, there is a 2D-ACG $\mathscr{G} \in \mathbf{G}_{\text{string,string}}(2,2,2)$ such that*

$$\mathcal{R}(T) = \mathcal{O}(\mathscr{G})$$

where $\mathcal{R}(T)$ denotes the relation defined by T.

Here we would like to emphasize that even though it is known that each of the domain and range of a relation is represented by an ACG, the question whether the relation itself can be represented by a 2D-ACG is not trivial. De Groote has shown Proposition 5.2 using Nivat's Theorem [39], which establishes the equivalence between finite state transducers and *bimorphisms*. A bimorphism $\mathscr{B}$ is a triple $\langle \mathcal{L}, \Phi, \Psi \rangle$ where $\mathcal{L}$ is a regular language, Φ and Ψ are homomorphisms. The relation defined by $\mathscr{B}$ is given as $\mathcal{R}(\mathscr{B}) = \{ \langle \Phi(\mathsf{w}), \Psi(\mathsf{w}) \rangle \mid \mathsf{w} \in \mathcal{L} \}$. We see the straightforward correspondence between homomorphisms and first-order lexicons from string signatures to string signatures. Together with the fact that regular languages are represented by ACGs belonging to $\mathbf{G}_{\text{string}}(2,2)$, this homomorphism-lexicon correspondence and Nivat's Theorem entail Proposition 5.2 (see Figure 5.1).

Tree bimorphisms are a tree-version of bimorphisms. A tree bimorphism $\mathscr{B} = \langle \mathcal{L}, \Phi, \Psi \rangle$ consists of a regular tree language $\mathcal{L}$ and two *tree homomorphisms* Φ and Ψ. Instead of giving the definition of tree bimorphisms in the usual way, it is enough to state just that a tree homomorphism is a first-order λK-lexicon[1] from a tree signature to a tree signature. The relation defined by $\mathscr{B}$ is given as $\mathcal{R}(\mathscr{B}) = \{ \langle \Phi(M), \Psi(M) \rangle \mid M \in \mathcal{L} \}$. As (string) bimorphisms are extended to tree bimorphisms, finite state transducers are extended to *tree transducers* (see [3] for instance). It is known that bottom-up tree transducers that are allowed to translate without reading the label on the current

[1] A λK-lexicon can map a constant to non-linear λ-terms. See Section 3.2.

node (*ε-rules*) correspond to tree-bimorphisms whose first homomorphisms Φ are semi-relabeling, and bottom-up tree transducers without ε-rules correspond to tree-bimorphisms whose first homomorphisms Φ are relabeling. Moreover, some kinds of restrictions, such as non-duplication, non-deletion, etc., on transducers and tree-homomorphisms preserve the equivalence. In particular, if a transducer is *linear* (non-duplicating and non-deleting), then Ψ is *linear* in the corresponding bimorphism. By the straightforward correspondence between tree-homomorphisms and lexicons, and between regular tree grammars and $\mathbf{G}_{\text{tree}}(2, 1(\text{r}))$ (Figure 5.1 is still valid for tree cases), we have the following corollary.

Corollary 5.3. *Every linear bottom-up tree transducer has an equivalent 2D-ACG belonging to $\mathbf{G}_{\text{tree,tree}}(2, 1(\text{sr}), 1)$, and vice versa. Every linear bottom-up tree transducer without ε-rules has an equivalent 2D-ACG belonging to $\mathbf{G}_{\text{tree,tree}}(2, 1(\text{r}), 1)$, and vice versa.*

Since linear bottom-up tree transducers and linear top-down tree transducers define the same class of tree languages, the above corollary holds for linear top-down transducers. We note that though all $\mathbf{G}_{\text{tree}}(2, 1(\text{r}))$, $\mathbf{G}_{\text{tree}}(2, 1(\text{sr}))$ and $\mathbf{G}_{\text{tree}}(2, 1)$ define the class of regular tree languages, we cannot drop the condition that "the first lexicon is (semi)relabeling" from the above statement.

Shieber [51] has shown that the relations defined by *synchronous tree substitution grammars (STSGs)* can be represented by *linear* tree bimorphisms, where a bimorphism is said to be linear iff both tree homomorphisms are linear. This immediately implies the following corollary.

Corollary 5.4. *Synchronous tree substitution grammars can be represented by 2D-ACGs belonging to $\mathbf{G}_{\text{tree,tree}}(2, 1, 1)$.*

In fact, Yamada [58] has given a direct encoding of STSGs by 2D-ACGs in $\mathbf{G}_{\text{tree,tree}}(2, 1, 1)$. The converse relation is open. Shieber has presented a method that constructs a "corresponding" STSG $G^{\mathscr{B}}$ for a given linear tree bimorphism $\mathscr{B} = \langle \mathcal{L}, \Phi, \Psi \rangle$ such that the tree relation defined by $\mathscr{B}$ is obtainable from the tree relation defined by the STSG by a relabeling homomorphism.

5.2 Examples

Proposition 5.5. *For every tree ACG $\mathscr{G} \in \mathbf{G}_{\text{tree}}(m, n)$, one can find a 2D-ACG $\mathscr{G}' \in \mathbf{G}_{\text{tree,string}}(m, n, n+1)$ such that*

$$\mathcal{O}(\mathscr{G}') = \{\, \langle P, \text{yield}(P) \rangle \mid P \in \mathcal{O}(\mathscr{G}) \,\}.$$

Proof. For $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$, let a string signature Σ_2 and a lexicon $\mathscr{L}_2 : \Sigma_1 \to \Sigma_2$ be defined by

$$\begin{aligned}
\mathscr{A}_2 &= \{o\}, \\
\mathscr{C}_2 &= \{\, \mathsf{c} \in \mathscr{C}_1 \mid \tau_1(\mathsf{c}) = o \,\}, \\
\tau_2(\mathsf{c}) &= \mathit{str} \text{ for all } \mathsf{c} \in \mathscr{C}_2, \\
\mathscr{L}'(o) &= \mathit{str} \\
\mathscr{L}'(\mathsf{c}) &= \begin{cases} \mathsf{c} & \text{if } \mathsf{c} \in \mathscr{C}_1 \cap \mathscr{C}_2 \\ \lambda x_1^{\mathit{str}} \dots x_n^{\mathit{str}}.x_1 + \dots + x_n & \text{if } \tau_1(\mathsf{c}) = o^n \to o \text{ with } n \geq 1. \end{cases}
\end{aligned}$$

The 2D-ACG $\mathscr{G}' = \langle \Sigma_0, \Sigma_1, \Sigma_2, \mathscr{L}, \mathscr{L}' \circ \mathscr{L}, s \rangle$ satisfies the desired condition. □

Proposition 5.6 (Reverse). *For every string ACG $\mathscr{G} \in \mathbf{G}_{\text{string}}(m, n)$, one can find a 2D-ACG $\mathscr{G}' \in \mathbf{G}_{\text{string,string}}(m, n, n+1)$ such that*

$$\mathcal{O}(\mathscr{G}') = \{\, \langle /\mathsf{w}/, /\mathsf{w}^R/ \rangle \mid /\mathsf{w}/ \in \mathcal{O}(\mathscr{G}) \,\}$$

where w^R denotes the reverse of w.

Proof. For $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$, let Σ_0' and Σ_1' be respectively the extensions of Σ_0 and Σ_1 such that

$$\Sigma_0' \begin{cases} \mathscr{A}_0' = \mathscr{A}_0 \cup \{s'\} \ (s' \notin \mathscr{A}_0), \\ \mathscr{C}_0' = \mathscr{C}_0 \cup \{\#\} \ (\# \notin \mathscr{C}_0), \\ \tau_0' = \tau_0 \cup \{\, \# \mapsto s \to s' \,\}, \end{cases} \qquad \Sigma_1' \begin{cases} \mathscr{A}_1' = \mathscr{A}_1 = \{o\}, \\ \mathscr{C}_1' = \mathscr{C}_1 \cup \{\#\} \ (\# \notin \mathscr{C}_1), \\ \tau_1' = \tau_1 \cup \{\, \# \mapsto o \,\}, \end{cases}$$

and define $\mathscr{L}'$ and $\mathscr{L}''$ as extensions of $\mathscr{L}$ by

$$\mathscr{L}' : \Sigma_0' \to \Sigma_1 \begin{cases} \mathscr{L}'(s') = \mathit{str}, \\ \mathscr{L}'(\#) = \lambda z^{\mathit{str}}.z, \end{cases} \qquad \mathscr{L}'' : \Sigma_0' \to \Sigma_1' \begin{cases} \mathscr{L}''(s') = o, \\ \mathscr{L}''(\#) = \lambda z^{\mathit{str}}.z\#. \end{cases}$$

Thus, the ACG $\langle \Sigma_0', \Sigma_1, \mathscr{L}', s' \rangle$ is equivalent to $\mathscr{G}$ and the ACG $\langle \Sigma_0', \Sigma_1', \mathscr{L}'', s' \rangle$ generates exactly $\mathcal{O}(\mathscr{G})$ with the endmarker $\#$.

Let a new second-order lexicon $\mathscr{L}^R : \Sigma_1' \to \Sigma_1$ be defined as

$$\begin{aligned}
\mathscr{L}^R(o) &= \mathit{str} \\
\mathscr{L}^R(\mathsf{a}) &= \lambda x^{\mathit{str}}.x + \mathsf{a} \\
\mathscr{L}^R(\#) &= /\varepsilon/.
\end{aligned}$$

We easily see that

$$\mathscr{L}^R(\mathsf{a}_1(\dots(\mathsf{a}_n\#)\dots)) = /\mathsf{a}_n \dots \mathsf{a}_1/.$$

Thus, for $\mathscr{G}' = \langle \Sigma'_0, \Sigma_1, \Sigma_1, \mathscr{L}', \mathscr{L}^R \circ \mathscr{L}'', s' \rangle$, we have

$$\mathcal{O}(\mathscr{G}') = \{\, \langle /\mathsf{w}/, /\mathsf{w}^R/ \rangle \mid /\mathsf{w}/ \in \mathcal{O}(\mathscr{G}) \,\}. \qquad \square$$

Suppose that all the elements of an alphabet $\mathscr{C}$ are ordered as $\langle \mathsf{c}_1, \ldots, \mathsf{c}_n \rangle$. The *sorted string* $\mathrm{Sort}(\mathsf{w})$ of a string $\mathsf{w} \in \mathscr{C}^*$ is defined as

$$\mathrm{Sort}(\mathsf{w}) = \mathsf{c}_1^{m_1} \ldots \mathsf{c}_n^{m_n}$$

where $m_i = \#_{\mathsf{c}_i}(\mathsf{w})$ for each $i \in \{1, \ldots, n\}$.

Proposition 5.7 (Sort). *For every string ACG $\mathscr{G} \in \mathbf{G}_{\mathrm{string}}(m, n)$, there is a 2D-ACG $\mathscr{G}' \in \mathbf{G}_{\mathrm{string,string}}(\max\{3, m\}, n, n)$ such that*

$$\mathcal{O}(\mathscr{G}') = \{\, \langle /\mathsf{w}/, /\mathrm{Sort}(\mathsf{w})/ \rangle \mid /\mathsf{w}/ \in \mathcal{O}(\mathscr{G}) \,\}.$$

Proof. Let $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$ and suppose that the elements of the object vocabulary $\mathscr{C}_1$ are ordered as $\langle \mathsf{c}_1, \ldots, \mathsf{c}_n \rangle$. We construct a 2D-ACG $\mathscr{G}' = \langle \Sigma'_0, \Sigma_1, \Sigma_1, \mathscr{L}', \mathscr{L}^S, s_n \rangle$ as follows. Let

$$\begin{aligned}
\mathscr{A}'_0 &= \mathscr{A}_0 \cup \{\, q_i, s_i \mid 1 \le i \le n \,\} \quad (q_i, s_i \notin \mathscr{A}_0), \\
\mathscr{C}'_0 &= \mathscr{C}_0 \cup \{\, \mathsf{B}_i, \mathsf{C}_i \mid 1 \le i \le n \,\} \quad (\mathsf{B}_i, \mathsf{C}_i \notin \mathscr{A}_0), \\
\tau'_0(\mathsf{a}) &= q_1^{k_1} \to \cdots \to q_n^{k_n} \to \tau_0(\mathsf{a}) \text{ for } k_i = \#_{\mathsf{c}_i}(\mathscr{L}(\mathsf{a})) \text{ for } \mathsf{a} \in \mathscr{C}_0 \\
\tau'_0(\mathsf{B}_i) &= s_{i-1} \to s_i \text{ where } s_0 = s, \\
\tau'_0(\mathsf{C}_i) &= (q_i \to s_i) \to s_i.
\end{aligned}$$

For $P \in \Lambda(\Sigma_1)$, let $\widetilde{P}$ denote the term obtained from P by replacing the j-th occurrence of each constant c_i with a fresh variable $x_{i,j}$, i.e., $\widetilde{P}$ is the constant-free term such that

$$P = \widetilde{P}[\mathsf{c}_i / x_{i,j}]_{1 \le i \le n}^{1 \le j \le k_i}$$

for the number k_i of the occurrences of c_i in P. We define

$$\begin{aligned}
\mathscr{L}'(p) = \mathscr{L}^S(p) &= \begin{cases} \mathscr{L}(p) & \text{if } p \in \mathscr{A}_0 \\ \mathit{str} & \text{otherwise} \end{cases} \\
\mathscr{L}'(\mathsf{a}) = \mathscr{L}^S(\mathsf{a}) &= \lambda x_{1,1}^{q_1} \ldots x_{1,k_1}^{q_1} \ldots x_{n,1}^{q_n} \ldots x_{n,k_n}^{q_n}.\widetilde{\mathscr{L}(\mathsf{a})} \\
&\qquad \text{for } k_i = \#_{\mathsf{c}_i}(\mathscr{L}(\mathsf{a})) \text{ for } \mathsf{a} \in \mathscr{C}_0 \\
\mathscr{L}'(\mathsf{B}_i) = \mathscr{L}^S(\mathsf{B}_i) &= \lambda x^{\mathit{str}}.x, \\
\mathscr{L}'(\mathsf{C}_i) &= \lambda x^{\mathit{str} \to \mathit{str}}.x/\mathsf{c}_i/. \\
\mathscr{L}^S(\mathsf{C}_i) &= \lambda x^{\mathit{str} \to \mathit{str}}.x/\varepsilon/ + /\mathsf{c}_i/.
\end{aligned}$$

We need to show

1. $\langle /\mathsf{w}_1/, /\mathsf{w}_2/\rangle \in \mathcal{O}(\mathscr{G}')$ implies $\#_{\mathsf{c}_i}(/\mathsf{w}_1/) = \#_{\mathsf{c}_i}(/\mathsf{w}_2/)$ for all $\mathsf{c}_i \in \mathscr{C}_1$,
2. $/\mathsf{w}/ \in \mathcal{O}_2(\mathscr{G}')$ implies $\mathsf{w} \in \{\mathsf{c}_1\}^* \dots \{\mathsf{c}_1\}^*$,
3. $\mathcal{O}(\mathscr{G}) = \mathcal{O}_1(\mathscr{G}')$,

(1) is obvious.

(2) If $N \in \mathcal{A}(\mathscr{G}')$, then N has the form

$$\begin{aligned} N = \mathsf{C}_n(\lambda x_{n,1}^{q_n}.\mathsf{C}_n(\lambda x_{n,2}^{q_n}.\dots\mathsf{C}_n(\lambda x_{n,m_n}^{q_n}.\mathsf{B}_n(\\ \vdots \\ \mathsf{C}_1(\lambda x_{1,1}^{q_1}.\mathsf{C}_1(\lambda x_{1,2}^{q_1}.\dots\mathsf{C}_1(\lambda x_{1,m_1}^{q_1}.\mathsf{B}_1 N')\dots))\dots))\dots)). \end{aligned}$$

Thus we see (2) as

$$\mathscr{L}^S(N) = \mathscr{L}^S(N')[/\varepsilon//x_{i,j}]_{1\le i\le n}^{1\le j\le m_i} + /\mathsf{c}_1^{m_1}/ + \dots + /\mathsf{c}_n^{m_n}/ = /\mathsf{c}_1^{m_1}\dots\mathsf{c}_n^{m_n}/.$$

(3) We show $\mathcal{O}(\mathscr{G}) \subseteq \mathcal{O}_1(\mathscr{G}')$. Suppose that $M \in \mathcal{A}(\mathscr{G})$ is given. Let k be the total number of all occurrences of constants in M. That is,

$$M = M'[\mathsf{a}_1/z_1, \dots, \mathsf{a}_k/z_k]$$

for a constant-free term M' with $\mathrm{Fv}(M') = \{z_1, \dots, z_k\}$. For $1 \le i \le n$ and $1 \le j \le k$, let

$$\begin{aligned} l_{j,i} &= \sum_{1\le h\le j} \#_{\mathsf{c}_i}(\mathscr{L}(\mathsf{a}_h)) \\ m_i &= l_{k,i} = \#_{\mathsf{c}_i}(\mathscr{L}(M)). \end{aligned}$$

Let

$$N' = M'[\mathsf{a}_1\langle x_{i,j}^{q_i}\rangle_{1\le i\le n}^{1\le j\le l_{1,i}}/z_1, \mathsf{a}_2\langle x_{i,j}^{q_i}\rangle_{1\le i\le n}^{l_{1,i}+1\le j\le l_{2,i}}/z_2, \dots, \mathsf{a}_k\langle x_{i,j}^{q_i}\rangle_{1\le i\le n}^{l_{k-1,i}+1\le j\le l_{k,i}}/z_k].$$

Indeed N' is well-typed on Σ_0' and

$$\mathscr{L}(M) = \mathscr{L}'(N')[\mathsf{c}_i/x_{i,j}]_{1\le i\le n}^{1\le j\le m_i}.$$

For

$$\begin{aligned} N = \mathsf{C}_n(\lambda x_{n,1}^{q_n}.\mathsf{C}_n(\lambda x_{n,2}^{q_n}.\dots\mathsf{C}_n(\lambda x_{n,m_n}^{q_n}.\mathsf{B}_n(\\ \vdots \\ \mathsf{C}_1(\lambda x_{1,1}^{q_1}.\mathsf{C}_1(\lambda x_{1,2}^{q_1}.\dots\mathsf{C}_1(\lambda x_{1,m_1}^{q_1}.\mathsf{B}_1 N')\dots))\dots))\dots)), \end{aligned}$$

we have $N \in \mathcal{A}(\mathscr{G}')$ and $\mathscr{L}'(N) = \mathscr{L}(M)$.

The converse direction $\mathcal{O}_1(\mathscr{G}') \subseteq \mathcal{O}(\mathscr{G})$ is shown exactly symmetrically. We easily see that every occurrence of a constant a in $N \in \mathcal{A}(\mathscr{G})$ accompanies variables as $\mathsf{a}\vec{x}_1 \dots \vec{x}_n$ where the length of each $\vec{x}_i$ is $\#_{\mathsf{c}_i}(\mathscr{L}(\mathsf{a}))$. □

Proposition 5.8 (Reordering). *For every string ACG* $\mathscr{G} \in \mathbf{G}_{\text{string}}(m,n)$*, there is a 2D-ACG* $\mathscr{G}' \in \mathbf{G}_{\text{string,string}}(\max\{3,m\},n,n)$ *such that*

$$\mathcal{O}(\mathscr{G}') = \{\, \langle /\mathsf{w}/, /\mathsf{v}/\rangle \mid /\mathsf{w}/ \in \mathcal{O}(\mathscr{G}),\ \#_{\mathsf{c}_i}(\mathsf{w}) = \#_{\mathsf{c}_i}(\mathsf{v}) \text{ for all } \mathsf{c}_i \in \mathscr{C}_1 \,\}.$$

Proof. Replace each s_i in the proof of Proposition 5.7 with $s = s_0$. □

5.3 One-Dimensionality and Two-Dimensionality

One may wonder whether a 1D-ACG can encode a 2D-ACG. Formally putting, given a 2D-ACG $\mathscr{G}$, can we construct an ACG $\mathscr{G}'$ such that

$$\mathcal{O}(\mathscr{G}') = \{\, \lambda w.wP_1P_2 \mid \langle P_1, P_2\rangle \in \mathcal{O}(\mathscr{G}) \,\}?$$

This is an open problem at this moment. We have just a partial answer to the question.

Proposition 5.9. *For every 2D-ACG* $\mathscr{G} \in \mathbf{G}(2,n_1,n_2)$*, there is an ACG* $\mathscr{G}' \in \mathbf{G}(2,n)$ *for* $n = \max\{n_1,n_2\}+2$ *such that*

$$\mathcal{O}(\mathscr{G}') = \{\, \lambda w.wP_1P_2 \mid \langle P_1, P_2\rangle \in \mathcal{O}(\mathscr{G}) \,\}.$$

Proof. Let $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \Sigma_2, \mathscr{L}_1, \mathscr{L}_2, s\rangle \in \mathbf{G}(2,n_1,n_2)$. We define the corresponding ACG $\mathscr{G}' \in \mathbf{G}(2,n)$ with $n = \max\{n_1,n_2\}+2$ to have the same abstract vocabulary as $\mathscr{G}$. The new lexicon $\mathscr{L}$ maps each atomic type p to

$$\mathscr{L}(p) = (\mathscr{L}_1(p) \to \mathscr{L}_2(p) \to o) \to o.$$

For each $\mathsf{a} \in \mathscr{C}_0$ of type $p_1 \to \cdots \to p_k \to q$, if $\mathscr{L}_1(\mathsf{a}) = \lambda x_1 \ldots x_k.P_1$ and $\mathscr{L}_2(\mathsf{a}) = \lambda y_1 \ldots y_k.P_2$, then we define $\mathscr{L}(\mathsf{a})$ as

$$\mathscr{L}(\mathsf{a}) = \lambda z_1 \ldots z_k w.z_1(\lambda x_1 y_1.z_2(\lambda x_2 y_2.\ldots z_k(\lambda x_k y_k.wP_1P_2)\ldots)).$$

It is easy to check that for every variable-free $M \in \Lambda(\Sigma_0)$ of an atomic type, it holds that

$$\mathscr{L}(M) = \lambda w^{\tau_1(\mathscr{L}_1(M))\to\tau_2(\mathscr{L}_2(M))\to o}.w\mathscr{L}_1(M)\mathscr{L}_2(M).$$ □

It is easy to extend the above proposition for second-order kD-ACGs for $k \geq 2$.

Corollary 5.10. *Every second-order kD-ACG* $\mathscr{G} \in \mathbf{G}(2,n_1,\ldots,n_k)$ *generates a PTIME language.*

Proof. By Proposition 5.9 and Theorem 2.11. □

Corollary 5.11. *For every finite state transducer T, there is a linear context-free rewriting system G^T such that*

$$\mathcal{L}(G^T) = \{\, \mathsf{w}_1 \# \mathsf{w}_2 \mid \langle \mathsf{w}_1, \mathsf{w}_2 \rangle \in \mathcal{R}(T) \,\}$$

where $\#$ is a new symbol not in T and $\mathcal{L}(G^T)$ denotes the language defined by G^T.

Proof. For a 2D-ACG $\mathscr{G}^T \in \mathbf{G}_{\text{string,string}}(2,2,2)$ representing the relation defined by the finite state transducer T, let $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle \in \mathbf{G}_{\text{string}}(2,4)$ be the one-dimensional representation of $\mathscr{G}^T$ constructed by Proposition 5.9.

$$\mathcal{O}(\mathscr{G}) = \{\, \lambda w.w/\mathsf{w}_1//\mathsf{w}_2/ \mid \langle \mathsf{w}_1, \mathsf{w}_2 \rangle \in \mathcal{R}(T) \,\}.$$

Let $\mathscr{G}' = \langle \Sigma_0', \Sigma_1', \mathscr{L}', s' \rangle \in \mathbf{G}_{\text{string}}(2,4)$ be the extension of $\mathscr{G}$ obtained by adding the lexical entry

$$\langle \#,\ s \to s',\ \lambda w.w(\lambda xy.x + \# + y) \rangle$$

where s' is a new abstract atomic type mapped to *str*. Then,

$$\mathcal{O}(\mathscr{G}') = \{\, /\mathsf{w}_1 \# \mathsf{w}_2/ \mid \langle \mathsf{w}_1, \mathsf{w}_2 \rangle \in \mathcal{R}(T) \,\}.$$

The equivalence between LCFRSs and $\mathbf{G}_{\text{string}}(2,4)$ (Theorem 2.6) completes the proof. □

5.4 Linear Macro Tree Transducers

One well-known extension of top-down tree transducers is *macro tree transducers (MTTs)* [2,6], which are obtained by adding the feature of context-free tree grammars to top-down tree transducers. Their output tree languages for fixed input form context-free tree languages. Even though RTGs and linear CFTGs are equivalent to ACGs belonging to $\mathbf{G}_{\text{tree}}(2,1)$ and $\mathbf{G}_{\text{tree}}(2,2)$ respectively, it is not obvious whether linear MTTs are also equivalent to $\mathbf{G}_{\text{tree,tree}}(2,1,2)$. In fact we show the following proposition in this section.

Theorem 5.12. *Linear ε-free macro tree transducers are equivalent to 2D-ACGs belonging to $\mathbf{G}_{\text{tree,tree}}(2,1(\text{r}),2)$. Linear macro tree transducers are equivalent to 2D-ACGs belonging to $\mathbf{G}_{\text{tree,tree}}(2,1(\text{sr}),2)$.*

Our translation method is essentially same as the proof of the correspondence between CFTGs and MTTs by Engelfriet and Vogler [7]. While the CFTG obtained by their construction from a given MTT may have infinitely many nonterminal symbols and production rules, the CFTG obtained from a given *linear* MTT through our method is a usual *linear* CFTG consisting of finitely many production rules.

Definition

In this section, we provide two fixed special disjoint sets of variables, $\mathscr{X} = \{x_1, \dots, \}$ and $\mathscr{Y} = \{y_1, \dots\}$, and let $\mathscr{X}_k = \{x_1, \dots, x_k\}$, $\mathscr{Y}_k = \{y_1, \dots, y_k\}$. For usual variables, we use letters $z, z_0, z_1, z_2, \dots \in \mathscr{Z}$.

For fixed ranked alphabets Q and Σ, let $RHS(Q, \Sigma, \mathscr{X}, \mathscr{Y}) \subseteq \mathbb{T}(Q \cup \Sigma \cup \mathscr{X} \cup \mathscr{Y})$ be the smallest set RHS such that

- $\mathscr{Y} \subseteq RHS$,
- if $\mathsf{f} \in \Sigma^{(m)}$ and $M_1, \dots, M_m \in RHS$, then $\mathsf{f} M_1 \dots M_m \in RHS$,
- if $q \in Q^{(1+n)}$, $x_i \in \mathscr{X}$, and $N_1, \dots, N_n \in RHS$, then $q x_i N_1 \dots N_n \in RHS$.

A *macro tree transducer* (MTT) is a quintuple $T = \langle \Sigma_1, \Sigma_2, Q, \Delta, q_s \rangle$ where

- Σ_1, Σ_2 and Q are ranked alphabets such that $(\Sigma_1 \cup \Sigma_2) \cap Q = \varnothing$ and $Q^{(0)} = \varnothing$,
- $q_s \in Q^{(1)}$ is called the *initial state*,
- Δ is the set of transition rules each of which is of the form
  - $q(\mathsf{f} x_1 \dots x_m) y_1 \dots y_n \to M$, or
  - $q x_1 y_1 \dots y_n \to N$ (called an *ε-rule*),

  where $\mathsf{f} \in \Sigma_1^{(m)}$, $q \in Q^{(1+n)}$, $M \in RHS(Q, \Sigma_2, \mathscr{X}_m, \mathscr{Y}_n)$ and $N \in RHS(Q, \Sigma_2, \mathscr{X}_1, \mathscr{Y}_n)$.

An MTT is *ε-free* if it has no ε-rule.

For $M, M' \in \mathbb{T}(Q \cup \Sigma_1 \cup \Sigma_2)$ and $\rho = q(\mathsf{f} x_1 \dots x_m) y_1 \dots y_n \to K \in \Delta$, we write $M \vdash_T^\rho M'$ iff there is $M_0 \in \mathbb{T}(Q \cup \Sigma_1 \cup \Sigma_2 \cup \{z\})$ such that

- z occurs exactly once in M_0,
- $M = M_0[q(\mathsf{f} M_1 \dots M_m) N_1 \dots N_n / z]$,
- $M' = M_0[K[M_1/x_1, \dots, M_m/x_m, N_1/y_1, \dots, N_n/y_n]/z]$.

For an ε-rule $\rho = q x_1 y_1 \dots y_n \to K \in \Delta$, we write $M \vdash_T^\rho M'$ iff there is $M_0 \in \mathbb{T}(Q \cup \Sigma_1 \cup \Sigma_2 \cup \{z\})$ such that

- z occurs exactly once in M_0,

- $M = M_0[qM_1N_1 \dots N_n/z]$,
- $M' = M_0[K[M_1/x_1, N_1/y_1, \dots, N_n/y_n]/z]$.

$\vdash_T$ is defined to be the union of $\vdash_T^\rho$ for all $\rho \in \Delta$ and $\vdash_T^*$ is the reflexive and transitive closure of $\vdash_T$. The relation $\mathcal{R}(T)$ determined by T is given as

$$\mathcal{R}(T) = \{\, \langle M, N\rangle \mid M \in \mathbb{T}(\Sigma_1),\ N \in \mathbb{T}(\Sigma_2), \text{ and } q_s M \vdash_T^* N \,\}.$$

We say that an MTT T is *linear* iff

- for every $q(\mathsf{f}x_1 \dots x_m)y_1 \dots y_n \to K \in \Delta$, each x_i and y_j appears in K exactly once for $1 \le i \le m$ and $1 \le j \le n$ and
- for every $qx_1y_1 \dots y_n \to K \in \Delta$, each x_1 and y_j appears in K exactly once for $1 \le j \le n$.

If T is linear, then for a rule $\rho = q(\mathsf{f}x_1 \dots x_m)y_1 \dots y_n \to K \in \Delta$, each x_i has the unique parent $q_i \in Q$ in K. That is, for some $K_0 \in \mathbb{T}(\Sigma_2 \cup \mathscr{Y} \cup \mathscr{Z})$ and $\vec{K}_i$ consisting of trees in $\mathbb{T}(\Sigma_2 \cup \mathscr{Y} \cup \mathscr{Z})$, K can be represented as

$$K = K_0[q_{k_1}x_{k_1}\vec{K}_{k_1}/z_1] \dots [q_{k_m}x_{k_m}\vec{K}_{k_m}/z_m]$$

where $\langle k_1, \dots, k_m\rangle$ is a permutation of $\langle 1, \dots, m\rangle$. Note that the above does not represent a simultaneous substitution, but a sequential substitution; e.g., z_2 may appear in $\vec{K}_{k_1}$. For instance, if

$$K \equiv \mathsf{a}\big(q_1x_1\mathsf{b}(\mathsf{a}(\mathsf{a}(q_2x_2\mathsf{b}))y_1)\big)(q_3x_3\mathsf{b}),$$

where $\mathsf{a} \in \Sigma_2^{(2)}$, $\mathsf{b} \in \Sigma_2^{(0)}$, $q_1 \in Q^{(3)}$, $q_2, q_3 \in Q^{(2)}$, then K can be represented as

$$K \equiv \mathsf{a}z_1z_3[q_1x_1\mathsf{b}(\mathsf{a}(\mathsf{a}z_2)y_1)/z_1][q_2x_2\mathsf{b}/z_2][q_3x_3\mathsf{b}/z_3].$$

Technical Lemmas

For some technical reason, we extend the definition of $\vdash_T \subseteq \mathbb{T}(Q \cup \Sigma_1 \cup \Sigma_2) \times \mathbb{T}(Q \cup \Sigma_1 \cup \Sigma_2)$ to define the relation on $\mathbb{T}(Q \cup \Sigma_1 \cup \Sigma_2 \cup \mathscr{Z}) \times \mathbb{T}(Q \cup \Sigma_1 \cup \Sigma_2 \cup \mathscr{Z})$ in the obvious way.

Definition 5.13. For fixed ranked alphabets Q, Σ_1, Σ_2, the set of *sentential forms* $SF(Q, \Sigma_1, \Sigma_2, \mathscr{Z}) \subseteq \mathbb{T}(Q \cup \Sigma_1 \cup \Sigma_2 \cup \mathscr{Z})$ is defined as the smallest set SF such that

- $\mathscr{Z} \subseteq SF$,

- if $\mathsf{f} \in \Sigma_2^{(m)}$ and $N_1, \ldots, N_m \in SF$, then $\mathsf{f}N_1 \ldots N_m \in SF$,
- if $q \in Q^{(1+n)}$, $M \in \mathbb{T}(\Sigma_1)$, and $N_1, \ldots, N_n \in SF$, then $qMN_1 \ldots N_n \in SF$.

We simply write SF for $SF(Q, \Sigma_1, \Sigma_2, \mathscr{Z})$ if Q, Σ_1, Σ_2 are obvious from the context.

Lemma 5.14 (Engelfriet and Vogler [7]). *Let $T = \langle \Sigma_1, \Sigma_2, Q, q_s, \Delta \rangle$ be an MTT. For all $M \in SF(Q, \Sigma_1, \Sigma_2, \mathscr{Z})$ and M', $M \vdash_T M'$ implies $M' \in SF(Q, \Sigma_1, \Sigma_2, \mathscr{Z})$.*

Clearly $q_s M \in SF$ for $M \in \mathbb{T}(\Sigma_1)$, so we can ignore trees that are not in the sentential form.

If $L \in SF$, then L can be represented as

$$L = K_0[q_1 M_1 \vec{K}_1/z_1] \ldots [q_m M_m \vec{K}_m/z_m]$$

where each $\vec{K}_i$ is a sequence of trees in $\mathbb{T}(\Sigma_2 \cup \mathscr{Z})$, $K_0 \in \mathbb{T}(\Sigma_2 \cup \mathscr{Z})$, $M_i \in \mathbb{T}(\Sigma_1)$ and each z_j with $j \in \{1, \ldots, m\}$ occurs in $K_0, \vec{K}_1, \ldots, \vec{K}_m$ exactly once. (We assume that $z_1, \ldots, z_m$ do not occur in L.)

Lemma 5.15. *Let $T = \langle \Sigma_1, \Sigma_2, Q, q_s, \Delta \rangle$ be a linear MTT. Suppose that $L \in SF$ is represented as*

$$L = z_1[P_1/z_1] \ldots [P_k/z_k]$$

where $P_i \in SF$, z_i occurs in $z_1, P_1, \ldots, P_{i-1}$ exactly once, and z_i does not occur in $P_i, \ldots, P_k$ for each $i \in \{1, \ldots, k\}$. If $L \vdash_T^\rho L'$, then there exist $j \in \{1, \ldots, k\}$ and $P'_j \in SF$ such that

1. *$P_j \vdash_T^\rho P'_j$,*
2. *$L' = z_1[P'_1/z_1] \ldots [P'_k/z_k]$ where $P'_i = P_i$ for $i \neq j$,*
3. *$P'_i \in SF$ for each $i \in \{1, \ldots, k\}$.*
4. *z_i occurs in $z_1, P'_1, \ldots, P'_{i-1}$ exactly once, and z_i does not occur in $P'_i, \ldots, P'_k$ for each $i \in \{1, \ldots, k\}$.*

Proof. If $L \vdash_T^\rho L'$, we have two cases.

Case 1. Suppose that ρ is not an ε-rule.

$$\rho = q(\mathsf{f}\vec{x})\vec{y} \to K$$

By the definition of a derivation, L and L' are written as

$$L = L_0[q(\mathtt{f}\vec{M})\vec{N}/z]$$
$$L' = L_0[K[\vec{M}/\vec{x}, \vec{N}/\vec{y}]/z].$$

Let j be such that the focused occurrence of q in L appears in P_j. Since z_j occurs exactly once in $z_1, P_1, \ldots, P_{j-1}$, the occurrence of q in P_j corresponds to exactly one occurrence of q in L. P_j can be represented as

$$P_j = P_{j,0}[q(\mathtt{f}\vec{M})\vec{N}'/z]$$

since $P_j \in SF$. For

$$P'_j = P_{j,0}[K[\vec{M}/\vec{x}, \vec{N}'/\vec{y}]/z],$$

we have $P_j \vdash_T^{\rho} P'_j$. (1) is proved. By Lemma 5.14, $P'_j \in SF$ and thus (3) is proved.

Let ϕ and ψ denote the sequential substitutions $[P_1/z_1] \ldots [P_{j-1}/z_{j-1}]$ and $[P_{j+1}/z_{j+1}] \ldots [P_k/z_k]$, respectively. Since z_j occurs in $z_1, P_1, \ldots, P_{j-1}$ exactly once, we have

$$L_0 = z_1\phi[P_{j,0}/z_j]\psi$$
$$\vec{N} = \vec{N}'\psi,$$

where each element of $z_{j+1}, \ldots, z_k$ appears in either $z_1\phi[P_{j,0}/z_j]$ or $\vec{N}'$. Therefore,

$$\begin{aligned} L' &= L_0[K[\vec{M}/\vec{x}, \vec{N}/\vec{y}]/z] \\ &= z_1\phi[P_{j,0}/z_j]\psi[K[\vec{M}/\vec{x}, \vec{N}/\vec{y}]/z] \\ &= z_1\phi[P_{j,0}/z_j][K[\vec{M}/\vec{x}, \vec{N}'/\vec{y}]/z]\psi \\ &= z_1\phi[P'_j/z_j]\psi. \end{aligned}$$

(2) is proved.

(4) is clear by the linearity.

Case 2. If ρ is an ε-rule, similarly we can show the lemma. □

Corollary 5.16. *Let $T = \langle \Sigma_1, \Sigma_2, Q, q_s, \Delta \rangle$ be a linear MTT. Suppose that $L \in SF$ is represented as*

$$L = z_1[P_1/z_1] \ldots [P_k/z_k]$$

where $P_i \in SF$, z_i occurs in $P_1, \ldots, P_{i-1}$ exactly once, and z_i does not occur in $P_i, \ldots, P_k$ for each $i \in \{1, \ldots, k\}$. If

$$L \vdash_T^{\rho_1} \cdots \vdash_T^{\rho_l} L',$$

there is a partition $\langle S_1, \ldots, S_k \rangle$ of $\{1, \ldots, l\}$ such that

- $S_i = \{n_{i,1}, \ldots, n_{i,m_i}\}$ *with* $n_{i,j} < n_{i,j+1}$,
- $P_i \vdash_{\rho_{n_{i,1}}} \cdots \vdash_{\rho_{n_{i,m_i}}} P'_i$,
- $L' = z_1[P'_1/z_1] \ldots [P'_k/z_k]$.

Transformation

Here we present a method of transforming a linear MTT $T = \langle \Sigma_1, \Sigma_2, Q, \Delta, q_s \rangle$ into a 2D-ACG $\mathscr{G}^T = \langle \Sigma_0, \Sigma_1, \Sigma_2, \mathscr{L}_1, \mathscr{L}_2, s \rangle$. The set of abstract atomic types is defined as the set of states, $\mathscr{A}_0 = Q$ and define $\mathscr{L}_1(q) = o$ and $\mathscr{L}_2(q) = o^n \to o$ for each $q \in Q^{(1+n)}$. For each rule $\rho \in \Delta$, we put the following lexical entries into $\mathscr{G}^T$:

For $\rho = q(\mathsf{f}x_1 \ldots x_m)y_1 \ldots y_n \to K_0[q_{k_1} x_{k_1} \vec{K}_{k_1}/z_1] \ldots [q_{k_m} x_{k_m} \vec{K}_{k_m}/z_m]$, where $\langle k_1, \ldots, k_m \rangle$ is a permutation of $\langle 1, \ldots, m \rangle$,

$$\begin{aligned} [\![\rho]\!] &\in \mathscr{C}_0, \\ \tau_0([\![\rho]\!]) &= q_1 \to \cdots \to q_m \to q, \\ \mathscr{L}_1([\![\rho]\!]) &= \mathsf{f}, \\ \mathscr{L}_2([\![\rho]\!]) &= \lambda x_1 \ldots x_m y_1 \ldots y_n . K_0[x_{k_1} \vec{K}_{k_1}/z_1] \ldots [x_{k_m} \vec{K}_{k_m}/z_m], \end{aligned}$$

and for an ε-rule $\rho = q x_1 y_1 \ldots y_n \to K_0[q_1 x_1 \vec{K}_1/z_1]$,

$$\begin{aligned} [\![\rho]\!] &\in \mathscr{C}_0, \\ \tau_0([\![\rho]\!]) &= q_1 \to q, \\ \mathscr{L}_1([\![\rho]\!]) &= \lambda z^o . z, \\ \mathscr{L}_2([\![\rho]\!]) &= \lambda x_1 y_1 \ldots y_n . K_0[x_1 \vec{K}_1/z_1]. \end{aligned}$$

Since T is linear, $\mathscr{L}_2([\![\rho]\!])$ is indeed a well-typed linear λ-term.

Lemma 5.17. $\mathcal{O}(\mathscr{G}^T) \subseteq \mathcal{R}(T)$

Proof. We show by induction on M that if $M \in \Lambda(\Sigma_0)$ is variable-free and has an atomic type $q \in Q^{(1+n)}$, then

$$q|\mathscr{L}_1(M)|_\beta N_1 \ldots N_n \vdash_T^* |\mathscr{L}_2(M) N_1 \ldots N_n|_\beta$$

for every $N_i \in \mathbb{T}(\Sigma_2 \cup \mathscr{Z})$. This implies that for any $M \in \mathcal{A}(\mathscr{G}^T)$, we have $q_s|\mathscr{L}_1(M)|_\beta \vdash_T^* |\mathscr{L}_2(M)|_\beta$.

Let the head of M be $[\![\rho]\!]$. We have two cases.

Case 1. Suppose that $\rho \in \Delta$ is not an ε-rule,

$$\rho = q(\mathsf{f}x_1 \ldots x_m)y_1 \ldots y_n \to K_0[q_{k_1} x_{k_1} \vec{K}_{k_1}/z_1] \ldots [q_{k_m} x_{k_m} \vec{K}_{k_m}/z_m],$$

where $\langle k_1, \ldots, k_m \rangle$ is a permutation of $\langle 1, \ldots, m \rangle$. By $\tau_0([\![\rho]\!]) = q_1 \to \cdots \to q_m \to q$, M has the form

$$M = [\![\rho]\!] M_1 \ldots M_m$$

for some M_i with $\tau_0(M_i) = q_i$ and $\tau_0(M) = q$. Applying the induction hypothesis to each M_i, we have

$$q_i |\mathscr{L}_1(M_i)|_\beta \vec{K}_i \vdash_T^* |\mathscr{L}_2(M_i)\vec{K}_i|_\beta. \tag{5.1}$$

Let $\vec{x} = x_1 \ldots x_m$, $\vec{y} = y_1 \ldots y_n$, $\vec{N} = N_1 \ldots N_n$. By the definition,

$$\begin{aligned}
&\mathscr{L}_1([\![\rho]\!]) = \mathsf{f} \\
&\mathscr{L}_2([\![\rho]\!]) = \lambda \vec{x}\vec{y}.K_0[x_{k_1}\vec{K}_{k_1}/z_1] \ldots [x_{k_m}\vec{K}_{k_m}/z_m].
\end{aligned}$$

Since

$$\begin{aligned}
\mathscr{L}_2(M)\vec{N} &= \mathscr{L}_2([\![\rho]\!])\mathscr{L}_2(M_1) \ldots \mathscr{L}_2(M_m)\vec{N} \\
&= (\lambda \vec{x}\vec{y}.K_0[x_{k_1}\vec{K}_{k_1}/z_1] \ldots [x_{k_m}\vec{K}_{k_m}/z_m])\mathscr{L}_2(M_1) \ldots \mathscr{L}_2(M_m)\vec{N} \\
&\twoheadrightarrow_\beta K_0[\mathscr{L}_2(M_{k_1})\vec{K}_{k_1}/z_1] \ldots [\mathscr{L}_2(M_{k_m})\vec{K}_{k_m}/z_m][\vec{N}/\vec{y}],
\end{aligned}$$

we have

$$\begin{aligned}
q\mathscr{L}_1(M)N_1 \ldots N_n &= q(\mathsf{f}|\mathscr{L}_1(M_1)|_\beta \ldots |\mathscr{L}_1(M_m)|_\beta)\vec{N} \\
&\vdash_T^\rho K_0[q_{k_1}|\mathscr{L}_1(M_{k_1})|_\beta \vec{K}_{k_1}/z_1] \ldots [q_{k_m}|\mathscr{L}_1(M_{k_m})|_\beta \vec{K}_{k_m}/z_m][\vec{N}/\vec{y}] \\
&\vdash_T^* K_0[|\mathscr{L}_2(M_{k_1})\vec{K}_{k_1}|_\beta/z_1] \ldots [|\mathscr{L}_2(M_{k_m})\vec{K}_{k_m}|_\beta/z_m][\vec{N}/\vec{y}] \\
&\qquad\qquad \text{(By (5.1))} \\
&=_\beta \mathscr{L}_2(M)N_1 \ldots N_n.
\end{aligned}$$

Case 2. Suppose that ρ is an ε-rule of the form

$$qx_1y_1 \ldots y_n \to K_0[q_1x_1\vec{K}_1].$$

Then, $\tau_0([\![\rho]\!]) = q_1 \to q$, $M = [\![\rho]\!]M_1$ for some M_1 with $\tau_0(M_1) = q_1$, and $\tau_0(M) = q$. Applying the induction hypothesis to M_1, we have

$$q_1|\mathscr{L}_1(M_1)|_\beta \vec{K}_1 \vdash_T^* |\mathscr{L}_2(M_1)\vec{K}_1|_\beta.$$

By the definition,

$$\begin{aligned}
&\mathscr{L}_1([\![\rho]\!]) = \lambda z^o.z \\
&\mathscr{L}_2([\![\rho]\!]) = \lambda x_1y_1 \ldots y_n.K_0[x_1\vec{K}_1/z_1].
\end{aligned}$$

Let $\vec{y} = y_1 \dots y_n$ and $\vec{N} = N_1 \dots, N_n$. Since

$$\begin{aligned}\mathscr{L}_2(M)\vec{N} &= \mathscr{L}_2([\![\rho]\!])\mathscr{L}_2(M_1)\vec{N} \\ &= (\lambda x_1 \vec{y}.K_0[x_1\vec{K}_1/z_1])\mathscr{L}_2(M_1)\vec{N} \\ &\twoheadrightarrow_\beta K_0[\mathscr{L}_2(M_1)\vec{K}_1/z_1][\vec{N}/\vec{y}],\end{aligned}$$

we have

$$\begin{aligned}q\mathscr{L}_1(M)\vec{N} &\twoheadrightarrow_\beta q|\mathscr{L}_1(M_1)|_\beta\vec{N} \\ &\vdash_T^\rho K_0[q_1|\mathscr{L}_1(M_1)|_\beta\vec{K}_1/z_1][\vec{N}/\vec{y}] \\ &\vdash_T^* K_0[|\mathscr{L}_2(M_1)\vec{K}_1|_\beta/z_1][\vec{N}/\vec{y}] \\ &=_\beta \mathscr{L}_2(M)\vec{N}.\end{aligned}$$

□

Lemma 5.18. $\mathcal{R}(T) \subseteq \mathcal{O}(\mathscr{G}^T)$.

Proof. We show by induction on k that if

$$qMN_1 \dots N_n \vdash_T^{\rho_0} P_1 \vdash_T^{\rho_1} \cdots \vdash_T^{\rho_k} P_k$$

where $q \in Q^{(1+n)}$, $M \in \mathbb{T}(\Sigma_1)$ and $N_1, \dots, N_n, P_k \in \mathbb{T}(\Sigma_2 \cup \mathscr{Z})$, then there is $L \in \Lambda(\Sigma_0)$ of type q such that $|\mathscr{L}_1(L)|_\beta \equiv M$ and $|\mathscr{L}_2(L)N_1 \dots N_n|_\beta \equiv P_k$.

Case 1. Suppose that ρ_0 is not an ε-rule. Then, M, ρ_0, P_1 can be written as

$$\begin{aligned}M &= \mathsf{f}M_1 \dots M_m \\ \rho_0 &= q(\mathsf{f}x_1 \dots x_m)y_1 \dots y_n \to K_0[q_{k_1}x_{k_1}\vec{K}_{k_1}/z_1] \dots [q_{k_m}x_{k_m}\vec{K}_{k_m}/z_m] \\ P_1 &= K_0[q_{k_1}M_{k_1}\vec{K}_{k_1}/z_1] \dots [q_{k_m}M_{k_m}\vec{K}_{k_m}/z_m][\vec{N}/\vec{y}]\end{aligned}$$

where $\vec{N} = N_1 \dots N_n$, $\vec{y} = y_1 \dots y_n$, and $\langle k_1, \dots, k_m\rangle$ is a permutation of $\langle 1, \dots, m\rangle$. By Lemma 5.16, we have a partition $\langle S_1, \dots, S_m\rangle$ of $\{1, \dots, k\}$ such that

- $S_i = \{n_{i,1}, \dots, n_{i,l_i}\}$ with $n_{i,j} < n_{i,j+1}$,
- $q_iM_i\vec{K}_i[\vec{N}/\vec{y}] \vdash_T^{\rho_{n_{i,1}}} \cdots \vdash_T^{\rho_{n_{i,l_i}}} R_i$,
- $P_k = K_0[R_{k_1}/z_1] \dots [R_{k_m}/z_m]$.

By the induction hypothesis, there is $L_i \in \Lambda(\Sigma_0)$ of type q_i such that $\mathscr{L}_1(L_i) = M_i$ and

$$\mathscr{L}_2(L_i)\vec{K}_i[\vec{N}/\vec{y}] = R_i. \tag{5.2}$$

For $L = [\![\rho_0]\!] L_1 \dots L_m$, we see that

$$\mathscr{L}_1(L) = \mathscr{L}_1([\![\rho_0]\!])\mathscr{L}_1(L_1)\dots\mathscr{L}_1(L_m) = \mathsf{f}M_1\dots M_m = M,$$

and

$$\begin{aligned}
&\mathscr{L}_2(L)N_1\dots N_n \\
&= \mathscr{L}_2([\![\rho_0]\!])\mathscr{L}_2(L_1)\dots\mathscr{L}_2(L_m)\vec{N} \\
&= (\lambda x_1\dots x_m\vec{y}.K_0[x_{k_1}\vec{K}_{k_1}/z_1]\dots[x_{k_m}\vec{K}_{k_m}/z_m])\mathscr{L}_2(L_1)\dots\mathscr{L}_2(L_m)\vec{N} \\
&= K_0[\mathscr{L}_2(L_{k_1})\vec{K}_{k_1}[\vec{N}/\vec{y}]/z_1]\dots[\mathscr{L}_2(L_{k_m})\vec{K}_{k_m}[\vec{N}/\vec{y}]/z_m] \\
&= K_0[R_{k_1}/z_1]\dots[R_{k_m}/z_m] \qquad \text{(by (5.2))} \\
&= P_k.
\end{aligned}$$

Case 2. Suppose that ρ_0 is an ε-rule. Then, ρ_0 and P_1 can be written as

$$\begin{aligned}
\rho_0 &= qx_1y_1\dots y_n \to K_0[q_1x_1\vec{K}_1/z_1] \\
P_1 &= K_0[q_1M\vec{K}_1/z_1][\vec{N}/\vec{y}]
\end{aligned}$$

where $\vec{N} = N_1\dots N_n$ and $\vec{y} = y_1\dots y_n$. There is $R_1 \in \mathbb{T}(\Sigma_2 \cup \mathscr{Z})$ such that

- $q_1M\vec{K}_1[\vec{N}/\vec{y}] \vdash_T^{\rho_1} \cdots \vdash_T^{\rho_k} R_1$,
- $P_k = K_0[R_1/z_1]$.

By the induction hypothesis, there is $L_1 \in \Lambda(\Sigma_0)$ of type q_1 such that $\mathscr{L}_1(L_1) = M$ and

$$|\mathscr{L}_2(L_1)\vec{K}_1[\vec{N}/\vec{y}]|_\beta = R_1.$$

For $L = [\![\rho_0]\!] L_1$, we see that

$$\mathscr{L}_1(L) = \mathscr{L}_1([\![\rho_0]\!])\mathscr{L}_1(L_1) = (\lambda z^o.z)M = M,$$

and

$$\begin{aligned}
&\mathscr{L}_2(L)N_1\dots N_n \\
&= \mathscr{L}_2([\![\rho_0]\!])\mathscr{L}_2(L_1)N_1\dots N_n \\
&= (\lambda x_1y_1\dots y_n.K_0[x_1\vec{K}_1/z_1])\mathscr{L}_2(L_1)N_1\dots N_n \\
&= K_0[\mathscr{L}_2(L_1)\vec{K}_1[\vec{N}/\vec{y}]/z_1] \\
&= K_0[R_1/z_1] \\
&= P_k.
\end{aligned}$$

□

Corollary 5.19. *Every linear ε-free macro tree transducer has an equivalent 2D-ACG belonging to* $\mathbf{G}_{\mathrm{tree,tree}}(2, 1(\mathrm{r}), 2)$. *Every linear macro tree transducers has an equivalent 2D-ACGs belonging to* $\mathbf{G}_{\mathrm{tree,tree}}(2, 1(\mathrm{sr}), 2)$.

It is not difficult to establish the converse direction. We define a linear MTT $T^{\mathscr{G}} = \langle \Sigma_1, \Sigma_2, Q, q_s, \Delta \rangle$ equivalent to a given 2D-ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \Sigma_2, \mathscr{L}_1, \mathscr{L}_2, s \rangle \in \mathbf{G}_{\mathrm{tree,tree}}(2, 1(\mathrm{sr}), 2)$. Let

$$Q^{(1+n)} = \{\, q \in \mathscr{A}_0 \mid \mathscr{L}_2(q) = o^n \to o \,\}$$

and $q_s = s$. For each $\mathsf{a} \in \mathscr{C}_0$ of type $q_1 \to \cdots \to q_m \to q$, if $\mathscr{L}_2(q)$ is n-ary, then put the transition rule

$$q(|\mathscr{L}_1(\mathsf{a})x_1 \dots x_m|_\beta)y_1 \dots y_n \to |\mathscr{L}_2(\mathsf{a})(q_1x_1) \dots (q_mx_m)y_1 \dots y_n|_\beta$$

into Δ. It is easy to check that the above method is exactly the inverse of our encoding of linear MTTs by ACGs. Thus the resultant linear MTT $T^{\mathscr{G}}$ is equivalent to the given 2D-ACG $\mathscr{G}$. This shows the following statement.

Theorem 5.20. *Every linear ε-free macro tree transducer has an equivalent 2D-ACG belonging to* $\mathbf{G}_{\mathrm{tree,tree}}(2, 1(\mathrm{r}), 2)$ *and vice versa. Every linear macro tree transducers has an equivalent 2D-ACGs belonging to* $\mathbf{G}_{\mathrm{tree,tree}}(2, 1(\mathrm{sr}), 2)$ *and vice versa.*

Synchronous Tree Adjoining Grammars and 2D-ACGs

The notion of tree bimorphisms can be extended by replacing each homomorphism with general mappings, as lexicons are allowed to be higher-order. Shieber [52] has shown that the relation defined by *synchronous tree adjoining grammars (STAGs)*, which are an extension of STSGs, can be represented by *tree bimappings* $\mathscr{B} = \langle \mathcal{L}, \Phi, \Psi \rangle$ where $\mathcal{L}$ is a regular tree language and both mappings Φ and Ψ are defined by linear *monadic* ε-free MTTs. In a monadic MTT, $Q = Q^{(1)} \cup Q^{(2)}$ holds. Actually what Theorem 5.20 states is that linear MTTs can be represented by *2D-ACGs*, not by *lexicons*. Nevertheless, the following Lemma entails that STAGs are also encodable by 2D-ACGs.

Lemma 5.21. *Let two 2D-ACGs* $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \Sigma_2, \mathscr{L}_1, \mathscr{L}_2, s \rangle \in \mathbf{G}(2, 1(\mathrm{r}), n)$ *and* $\mathscr{G}' = \langle \Sigma'_0, \Sigma_1, \Sigma'_2, \mathscr{L}'_1, \mathscr{L}'_2, s' \rangle \in \mathbf{G}(2, 1(\mathrm{r}), n')$ *are given. There is a 2D-ACG* $\mathscr{G}'' \in \mathbf{G}(2, n, n')$ *such that*

$$\mathcal{O}(\mathscr{G}'') = \{\, \langle P, Q \rangle \mid \langle M, P \rangle \in \mathcal{O}(\mathscr{G}) \text{ and } \langle M, Q \rangle \in \mathcal{O}(\mathscr{G}') \,\}.$$

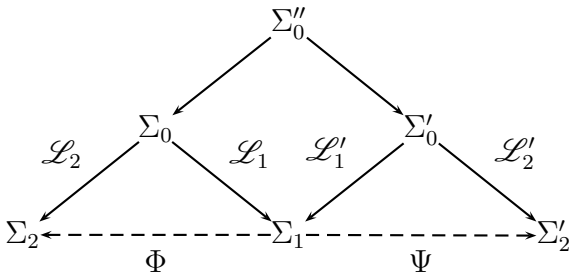


Figure 5.2: Bimapping and ACG (Lemma 5.21)

Proof. Let Σ''_0, $\mathscr{L}''_1 : \Sigma''_0 \to \Sigma_1$, $\mathscr{L} : \Sigma''_0 \to \Sigma_2$, $\mathscr{L}' : \Sigma''_0 \to \Sigma'_2$ be defined as

$$\begin{aligned}
\mathscr{A}''_0 &= \{\, [p,q] \mid p \in \mathscr{A}_0,\ q \in \mathscr{A}'_0,\ \mathscr{L}_1(p) = \mathscr{L}'_1(q) \,\},\\
\mathscr{C}''_0 &= \{\, [\![\mathsf{a},\mathsf{b}]\!] \mid \mathsf{a} \in \mathscr{C}_0,\ \mathsf{b} \in \mathscr{C}'_0,\ \mathscr{L}_1(\mathsf{a}) = \mathscr{L}'_1(\mathsf{b}) \,\},\\
\tau''_0([\![\mathsf{a},\mathsf{b}]\!]) &= [p_1,q_1] \to \cdots \to [p_k,q_k] \to [p_0,q_0]\\
&\qquad \text{if } \tau_0(\mathsf{a}) = p_1 \to \cdots \to p_k \to p_0,\ \tau_0(\mathsf{b}) = q_1 \to \cdots \to q_k \to q_0,\\
\mathscr{L}([p,q]) &= \mathscr{L}_2(p), \quad \mathscr{L}'([p,q]) = \mathscr{L}'_2(q),\\
\mathscr{L}([\![\mathsf{a},\mathsf{b}]\!]) &= \mathscr{L}_2(\mathsf{a}), \quad \mathscr{L}'([\![\mathsf{a},\mathsf{b}]\!]) = \mathscr{L}'_2(\mathsf{b}).
\end{aligned}$$

For $\mathscr{G}'' = \langle \Sigma''_0, \Sigma_2, \Sigma'_2, \mathscr{L}, \mathscr{L}', [s,s'] \rangle$, we have

$$\mathcal{O}(\mathscr{G}'') = \{\, \langle P, Q \rangle \mid \langle M, P \rangle \in \mathcal{O}(\mathscr{G}) \text{ and } \langle M, Q \rangle \in \mathcal{O}(\mathscr{G}') \,\}. \qquad \square$$

Corollary 5.22. *Synchronous tree adjoining grammars can be represented by 2D-ACGs belonging to* $\mathbf{G}_{\text{tree,tree}}(2, 2(\text{mon}), 2(\text{mon}))$.

Proof. For a given STAG G, let $\mathscr{B}^G = \langle \mathcal{L}, T_1, T_2 \rangle$ be the corresponding bimapping where T_1 and T_2 are linear monadic ε-free MTTs. By our construction, we have $\mathscr{G}^{T_i} \in \mathbf{G}_{\text{tree,tree}}(2, 1(\text{r}), 2(\text{mon}))$ such that

$$\mathcal{O}(\mathscr{G}^{T_i}) = \{\, \langle M, N \rangle \mid M \vdash_{T_i} N \,\}$$

By applying Lemma 5.21 to $\mathscr{G}^{T_1}$ and $\mathscr{G}^{T_2}$, we get $\mathscr{G} \in \mathbf{G}_{\text{tree,tree}}(2, 2(\text{mon}), 2(\text{mon}))$ such that

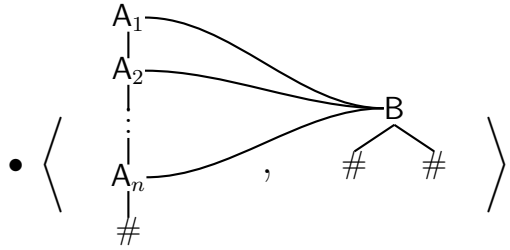
$$\mathcal{O}(\mathscr{G}) = \{\, \langle N_1, N_2 \rangle \mid M \vdash_{T_1} N_1,\ M \vdash_{T_2} N_2 \,\} = \mathcal{R}(\mathscr{B}^G).$$

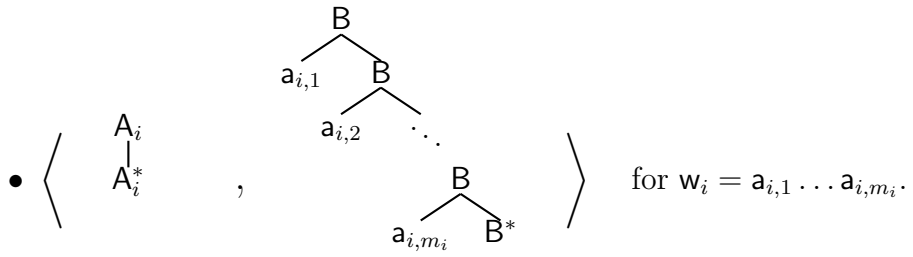
$\square$

Remark 5.23. Though STAGs generate PTIME languages by Corollaries 5.10 and 5.22, the universal membership problem for STAGs is NP-Complete. The membership of NP is obvious. The NP-hardness can be

shown by reduction from the string rearrangement problem (see Definition 6.4 and Theorem 6.5). For the definition of STAGs and their derivations, see [50].

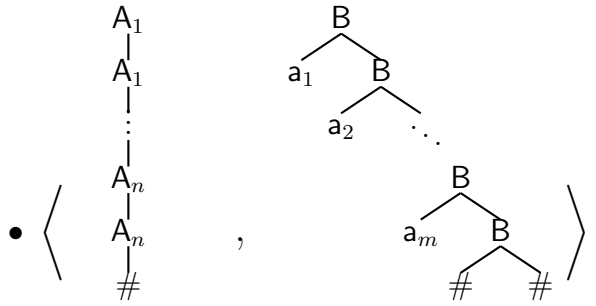
For an instance $\langle \mathsf{w}, \langle \mathsf{w}_1, \ldots, \mathsf{w}_n \rangle \rangle$ of the string rearrangement problem, let an STAG consists of the following initial tree pair and auxiliary tree pairs:





for $\mathsf{w}_i = \mathsf{a}_{i,1} \ldots \mathsf{a}_{i,m_i}$.

It is clear that $\langle \mathsf{w}_1, \ldots, \mathsf{w}_n \rangle$ has a solution iff the following tree pair is derivable by the above STAG. The order of adjoining trees corresponds to the permutation on $\langle \mathsf{w}_1, \ldots, \mathsf{w}_n \rangle$.



for $\mathsf{w} = \mathsf{a}_1 \ldots \mathsf{a}_m$.

This result makes a contrast to the fact that the universal membership problem for TAGs is in P.

5.5 Deterministic Tree Walking Transducers

A *deterministic tree walking transducer* [1] (DTWT) is a kind of finite state automaton that takes a tree as input and walks on the tree from a node to a node starting from the root node. Depending on the current state and the label on the current node, it deterministically decides which adjacent node it

should go to and which state it should enter. Each transition, it writes some string. When it gets out of the tree with a final state, the concatenation of the written strings is output. Thus, a DTWT defines a relation between trees and strings. A DTWT becomes a tree recognizer or string generator if we disregard its output or input, respectively. It is known that DTWTs as tree recognizers define a proper subclass of the regular tree languages [23] and that DTWTs as string generators are equivalent to LCFRSs [57]. Consequently it would be expected that the relation defined by DTWTs are definable by 2D-ACG belonging to $\mathbf{G}_{\mathrm{tree,string}}(2,1,4)$, though it is not a corollary to the above results. In this section, we show that the expectation is correct by borrowing the idea from Weir's work [57].

Definitions and Notations

For each tree M, we give the set $\mathrm{path}(M) \subseteq \mathbb{N}^*$ of *paths* of M as

$$\mathrm{path}(\mathsf{f}M_1 \dots M_n) = \{\varepsilon\} \cup \{\, i\pi \mid \pi \in \mathrm{path}(M_i) \,\}$$

and the subtree specified by $\pi \in \mathbb{N}^*$ is defined as

$$\mathsf{f}M_1 \dots M_n : \pi = \begin{cases} \mathsf{f}M_1 \dots M_n & \text{if } \pi = \varepsilon \\ M_i : \pi' & \text{if } \pi = i\pi'. \end{cases}$$

A *deterministic tree walking transducer* is a sextuple $T = \langle \Sigma_1, V, Q, \Delta, q_s, F \rangle$ where

- Σ_1 is a ranked alphabet, called *the input alphabet*,
- V is an unranked alphabet, called *the output alphabet*,
- Q is the set of *states*,
- $q_s \in Q$ is the *initial state*,
- $Q_f \subseteq Q$ is the set of *final states*,
- Δ is a partial function from $\Sigma_1 \times Q$ to $V^* \times \mathbb{N} \times Q$, called the set of *transition rules*, such that if $\Delta(\mathsf{A}, q) = \langle \mathsf{w}, d, q' \rangle$, then $0 \leq d \leq \mathrm{rank}(\mathsf{A})$.

A configuration of T is a quadruple $\langle K, \pi, q, \mathsf{w} \rangle$ where $K \in \mathbb{T}(\Sigma_1)$ is the tree under consideration, $\pi \in \mathrm{path}(K) \cup \{\uparrow\}$ specifies a node in K or is $\uparrow$ (where $\uparrow$ can be thought of as the parent of the root of K), $q \in Q$ is the current state, and $\mathsf{w} \in V^*$ is the output string produced up to that point in the computation. We call a state q an *up-state for* A if $\Delta(\mathsf{A}, q) = \langle \mathsf{v}, 0, q' \rangle$ for

some q' and v and a *down-state for* A if $\Delta(\mathsf{A}, q) = \langle \mathsf{v}, d, q' \rangle$ for some $d > 0$. We write

$$\langle K, \pi, q, \mathsf{w} \rangle \vdash_T \langle K, \pi', q', \mathsf{ww}' \rangle$$

if

- the head of $K : \pi$ is $\mathsf{A} \in \Sigma_1$,
- $\Delta(\mathsf{A}, q) = \langle \mathsf{w}', d, q' \rangle$,
- either $\pi = \pi' m$ for some m or $\pi = \varepsilon \wedge \pi' = \uparrow$ when $d = 0$,
- $\pi' = \pi d$ when $d > 0$.

The relation defined by T is

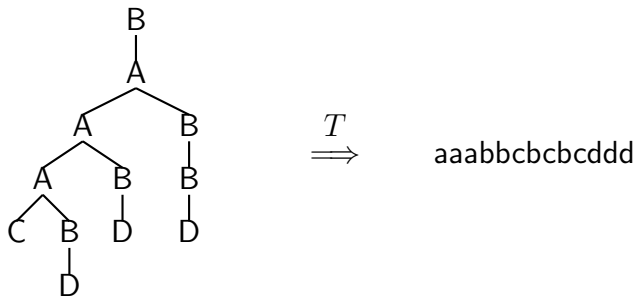
$$\mathcal{R}(T) = \{\, \langle K, \mathsf{w} \rangle \mid K \in \mathbb{T}(\Sigma_1) \text{ and } \langle K, \varepsilon, q_s, \varepsilon \rangle \vdash_T^* \langle K, \uparrow, q, \mathsf{w} \rangle \text{ for some } q \in F \,\}.$$

If K is in the domain of $\mathcal{R}(T)$, we say that K is *accepted*.

Example 5.24. Let a DTWT $T = \langle \Sigma_1, V, Q, \Delta, q_0, \{q_f\} \rangle$ have the following transition rules in Δ:

$$\begin{array}{llll}
\langle \mathsf{B}, q_0 \rangle \xrightarrow{\varepsilon} \langle 1, q_0 \rangle, & \langle \mathsf{A}, q_0 \rangle \xrightarrow{\mathsf{a}} \langle 1, q_0 \rangle, & \langle \mathsf{C}, q_0 \rangle \xrightarrow{\varepsilon} \langle 0, q_1 \rangle, & \langle \mathsf{A}, q_1 \rangle \xrightarrow{\varepsilon} \langle 0, q_1 \rangle, \\
\langle \mathsf{B}, q_1 \rangle \xrightarrow{\varepsilon} \langle 1, q_2 \rangle, & \langle \mathsf{A}, q_2 \rangle \xrightarrow{\varepsilon} \langle 2, q_2 \rangle, & \langle \mathsf{B}, q_2 \rangle \xrightarrow{\varepsilon} \langle 1, q_2 \rangle, & \langle \mathsf{D}, q_2 \rangle \xrightarrow{\varepsilon} \langle 0, q_3 \rangle, \\
\langle \mathsf{B}, q_3 \rangle \xrightarrow{\mathsf{b}} \langle 0, q_3 \rangle, & \langle \mathsf{A}, q_3 \rangle \xrightarrow{\mathsf{c}} \langle 1, q_2 \rangle, & \langle \mathsf{C}, q_2 \rangle \xrightarrow{\varepsilon} \langle 0, q_4 \rangle, & \langle \mathsf{A}, q_4 \rangle \xrightarrow{\mathsf{d}} \langle 0, q_4 \rangle, \\
\langle \mathsf{B}, q_4 \rangle \xrightarrow{\varepsilon} \langle 0, q_f \rangle & & &
\end{array}$$

T translates the following tree into the string aaabbcbcbcddd.



Encoding of DTWTs by 2D-ACGs

First we construct a 1D-ACG $\mathscr{G}_1 = \langle \Sigma_0, \Sigma_1, \mathscr{L}_1, s \rangle$ whose language is the domain of the relation defined by a given DTWT, disregarding the output strings. Suppose that a tree K is accepted by T. We can decorate each node of K with the sequence of states of T which T enters when T visits that

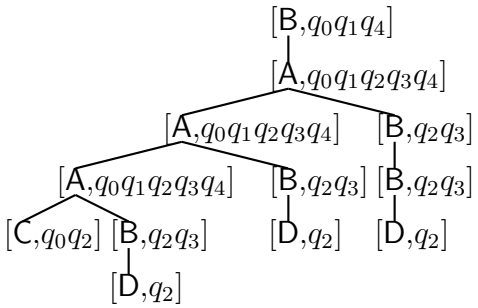


Figure 5.3: Tree decorated with valid histories

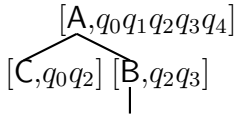
node. The tree in Example 5.24 is decorated as in Figure 5.3 according to the walking of T. We can give any decoration on a tree, but at most one of them represents the real walking of the automaton on the tree, since the automaton is deterministic. We call such a decoration *correct*. A tree admits exactly one correct decoration if the tree is accepted, and otherwise, it has no correct decoration. Though checking the correctness of a decoration on a tree can be done by walking the whole tree, here we present a method of checking the correctness of a decoration *locally*.

If a node labeled with A in a tree $K \in \mathbb{T}(\Sigma_1)$ is decorated with a sequence $\vec{p}$ of states, then $\vec{p}$ must end with an up-state p for A, since T terminates at the upper node $\uparrow$ of the root node. Moreover, by the determinacy of T, $\vec{p}$ contains no overlapped occurrences of the same state. If a state p appears in $\vec{p}$ twice or more, then T falls into a loop. We call a possibly empty sequence $\vec{p}$ of states *a valid history of* A if

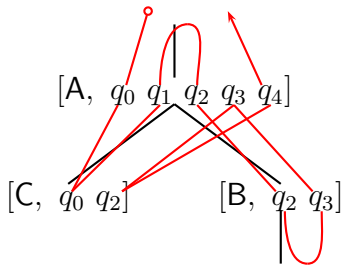
- for every element p of $\vec{p}$, $\Delta(\mathsf{A}, p)$ is defined,
- no state occurs twice or more in the sequences,
- the last element (if exists) of $\vec{p}$ is an up-state.

In particular, if A has rank 0, then every valid history $\vec{p}$ on A consists of up-states only. There are finitely many valid histories of A for each A, since any state can appear in a valid history at most once.

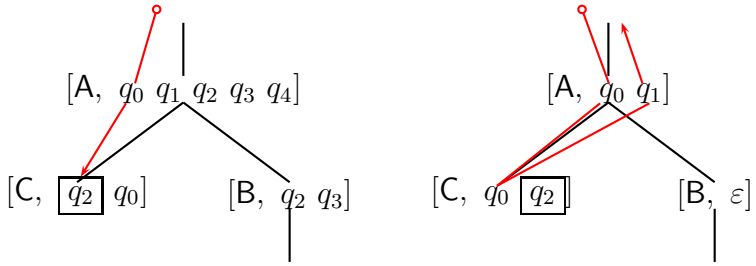
Let us look at the following fragment of the decorated tree in Figure 5.3.



Now you can forget other parts of the tree. Here $q_0q_1q_2q_3q_4$, q_0q_2, q_2q_3 are respectively valid histories of A, C, B. Since the automaton starts from the root node, the automaton visits the parent node A before its children C, B. Let us analyze the relation between those three valid histories starting from the state q_0 on the node A. According to the transition rule $\langle \mathsf{A}, q_0\rangle \xrightarrow{\mathsf{a}} \langle 1, q_0\rangle \in \Delta$, the automaton must go down to the first child C and the state q_0 is unchanged. Indeed, the first state of the decoration on the node C is q_0. It is consistent. Then the transition rule $\langle \mathsf{C}, q_0\rangle \xrightarrow{\varepsilon} \langle 0, q_1\rangle$ demands the automaton to go up to the parent A and to change the state into q_1. Indeed, the second state on A is q_1. Here the automaton should obey the rule $\langle \mathsf{A}, q_1\rangle \xrightarrow{\varepsilon} \langle 0, q_1\rangle$, and it disappears from our view. Though we cannot know whether the automaton comes back here unless we look at other parts of the tree, we assume that it comes back here with the third state q_2 on A, and continue checking the consistency as described so far. By $\langle \mathsf{A}, q_2\rangle \xrightarrow{\varepsilon} \langle 2, q_2\rangle$, the automaton arrives at B with the state q_2. Then it next goes down by the rule $\langle \mathsf{B}, q_2\rangle \xrightarrow{\varepsilon} \langle 1, q_2\rangle$. Again we assume the automaton comes back here with the next state q_3 on B, though we do not known whether the automaton really comes back here with that state. This consistency checking is illustrated by arrows as below, where all the states appearing in the valid histories are connected in sequence. This case we say it is a *locally consistent combination of valid histories*.



The following two are examples of *inconsistent combinations of valid histories*.



In the example on the left, though the sequence q_2q_0 is a valid history of C, it contradicts the transition rule $\langle \mathsf{A}, q_0\rangle \xrightarrow{\mathsf{a}} \langle 1, q_0\rangle$. In the example on the right, we cannot reach the last state q_2 on the node C. These are inconsistent combinations of valid histories. For $\mathsf{A} \in \Sigma_1^{(0)}$, which cannot have children, we also define a *locally consistent combination of valid histories* to be a valid history of A.

Let us turn our attention to the root node. It is easy to see that if the root node of an accepted tree is decorated as $[\mathsf{A}, q_1 \dots q_k]$, then

- the first element q_1 must be the initial state,
- if $i \neq k$, then q_i is a down state of A,
- $\langle \mathsf{A}, q_k\rangle \xrightarrow{\mathsf{v}} \langle 0, q_f\rangle \in \Delta$ for some $q_f \in Q_f$.

We call a valid history $\vec{q}$ on A satisfying those conditions a *root history of* A. As we will prove later, for a given decorated input tree, if each fragment of the tree of height one is decorated by consistent history and the root node is labeled with a root history, then the whole decoration on the tree is correct, and thus the undecorated tree is accepted by the automaton. For a given DTWT T, we construct a 2D-ACG whose abstract language represents the set of correctly decorated trees on Σ_1 and whose first lexicon strips off the decoration of the tree.

Definition 5.25. For $\mathsf{A} \in \Sigma_1^{(n)}$ and $\mathsf{B}_1, \dots, \mathsf{B}_n \in \Sigma_1$, a *locally consistent combination of valid histories of* $\langle \mathsf{A}, \mathsf{B}_1, \dots, \mathsf{B}_n\rangle$ is a sequence σ of pairs whose first component is a symbol in $\{\mathsf{A}, \mathsf{B}_1, \dots, \mathsf{B}_n\}$ and whose second component is a state in Q such that

$$\begin{aligned}\sigma &= \sigma_1 \dots \sigma_k \\ \sigma_i &= \sigma_{i,1} \dots \sigma_{i,k_i} \langle \mathsf{A}, p_{i,k_i+1}\rangle \\ \sigma_{i,j} &= \langle \mathsf{A}, p_{i,j}\rangle \langle \mathsf{B}_{d_{i,j}}, q_{i,j,1}\rangle \dots \langle \mathsf{B}_{d_{i,j}}, q_{i,j,k_{i,j}+1}\rangle\end{aligned}$$

where

- the sequence $\vec{p} = p_{1,1} \dots p_{1,k_1+1} \dots p_{k,1} \dots p_{k,k_k+1}$ is a valid history of A for each $1 \leq i \leq k$,
- $\vec{q}_h = \langle\, q_{i,j,g} \mid d_{i,j} = h,\ 1 \leq g \leq k_{i,j} + 1 \,\rangle$[2] (elements of $\vec{q}_h$ are paired with B_h) is a valid history of B_h

[2] Precisely, $\vec{q}_h$ is the sequence consisting of $q_{i,j,g}$ with $d_{i,j} = h$ such that if $\vec{q}_h = \vec{r}_1 q_{i_1,j_1,g_1} q_{i_2,j_2,g_2} \vec{r}_2$, then either $i_1 < i_2$ or $i_1 = i_2 \wedge j_1 < j_2$ or $i_1 = i_2 \wedge j_1 = j_2 \wedge g_1 < g_2$ holds. In the remainder of this section, if a sequence is defined without specifying the order of the elements, they are ordered in this manner according to their indexes.

and moreover, for each $1 \leq i \leq k$, $1 \leq j \leq k_i$, $1 \leq g \leq k_{i,j}$,

- $\Delta(\mathsf{A}, p_{i,k_i+1}) = \langle \mathsf{v}, 0, r \rangle$ for some $\mathsf{v} \in V^*$ and $r \in Q$,
- $\Delta(\mathsf{A}, p_{i,j}) = \langle \mathsf{v}_{i,j}, d_{i,j}, q_{i,j,1} \rangle$ with $1 \leq d_{i,j} \leq n$ and $\mathsf{v}_{i,j} \in V^*$,
- $\Delta(\mathsf{B}_{d_{i,j}}, q_{i,j,g}) = \langle \mathsf{v}, d, r \rangle$ for some $\mathsf{v} \in V^*$, $d > 0$, and $r \in Q$,
- $\Delta(\mathsf{B}_{d_{i,j}}, q_{i,j,k_{i,j}+1}) = \langle \mathsf{v}'_{i,j}, 0, p_{i,j+1} \rangle$ with $\mathsf{v}'_{i,j} \in V^*$.

Now we define a higher-order signature Σ_0 and a lexicon $\mathscr{L}_1 : \Sigma_0 \to \Sigma_1$. Let

$$\begin{aligned}\mathscr{A}_0 &= \{\, [\mathsf{A}, \vec{p}] \mid \vec{p} \text{ is a valid history of } \mathsf{A} \,\} \cup \{s\} \\ \mathscr{C}_0 &= \{\, [\![\mathsf{A}\mathsf{B}_1 \dots \mathsf{B}_n, \sigma]\!] \mid \sigma \text{ is a locally consistent combination of valid histories of } \mathsf{A}\mathsf{B}_1 \dots \mathsf{B}_n \,\} \\ &\quad \cup \{\, [\![\mathsf{A}, \vec{q}]\!] \mid \vec{q} \text{ is a root history of } \mathsf{A} \,\}\end{aligned}$$

and τ_0 be defined as

$$\tau_0([\![\mathsf{A}\mathsf{B}_1 \dots \mathsf{B}_n, \sigma]\!]) = [\mathsf{B}_1, \vec{q}_1] \to \dots \to [\mathsf{B}_n, \vec{q}_n] \to [\mathsf{A}, \vec{p}]$$

where each $\vec{q}_i$ and $\vec{p}$ are determined as in Definition 5.25,

$$\tau_0([\![\mathsf{A}, \vec{q}]\!]) = [\mathsf{A}, \vec{q}] \to s.$$

Let

$$\begin{aligned}\mathscr{L}_1([\mathsf{A}, \vec{q}]) &= o, \\ \mathscr{L}_1([\![\mathsf{A}\vec{\mathsf{B}}, \sigma]\!]) &= \mathsf{A}, \\ \mathscr{L}_1([\![\mathsf{A}, \vec{q}]\!]) &= \lambda z^o.z,\end{aligned}$$

We have established the correspondence between correctly decorated trees on $\mathbb{T}(\Sigma_1)$ and the abstract language of $\mathscr{G}_1 = \langle \Sigma_0, \Sigma_1, \mathscr{L}_1, s \rangle$.

Now, let turn our attention to the output of the transducer. We say that a valid history $\vec{q}$ on A is a *unitary history* iff $\vec{q}$ contains exactly one up-state for A. If a valid history $\vec{q}$ contains k up-states, then $\vec{q}$ is uniquely partitioned into k unitary histories. Suppose that a subtree K of a correctly decorated tree is rooted by $[\mathsf{A}, \vec{q}]$ such that $\vec{q}$ is partitioned into k unitary histories. This means that the automaton visits the subtree K exactly k times. The automaton outputs some string w_i during the walking of the i-th visit. Let $M \in \Lambda(\Sigma_0)$ represent K. We want the second lexicon $\mathscr{L}_2 : \Sigma_0 \to \Sigma_V$ to map M to

$$\lambda x^{\mathit{str}^k \to \mathit{str}}.x/\mathsf{w}_1/\dots/\mathsf{w}_k/.$$

This can be realized by defining $\mathscr{L}_2$ as follows.

$$\begin{aligned}\mathscr{L}_2(s) &= \mathit{str},\\ \mathscr{L}_2([\mathsf{A},\vec{p}]) &= (\mathit{str}^k \to \mathit{str}) \to \mathit{str}\\ &\qquad \text{if } \vec{p} \text{ is partitioned into } k \text{ unitary histories of } \mathsf{A}.\end{aligned}$$

Let $\mathsf{A}, \mathsf{B}_1, \ldots, \mathsf{B}_n$, and σ be as in Definition 5.25.

$$\mathscr{L}_2([\![\mathsf{A}\mathsf{B}_1 \ldots \mathsf{B}_n, \sigma]\!]) = \lambda y_1 \ldots y_n x.y_1(\lambda \vec{z}_1.y_2(\lambda \vec{z}_2. \ldots y_n(\lambda \vec{z}_n.xN_1 \ldots N_k)\ldots))$$

where $\vec{z}_h = \langle\, z_{i,j} \mid d_{i,j} = h \,\rangle$ and

$$N_i = /\mathsf{v}_{i,1}/ + z_{i,1} + /\mathsf{v}'_{i,1}/ + \cdots + /\mathsf{v}_{i,k_i}/ + z_{i,k_i} + /\mathsf{v}'_{i,k_i}/.$$

For root histories, let

$$\mathscr{L}_2([\![\mathsf{A},\vec{q}]\!]) = \lambda y^{(\mathit{str}\to\mathit{str})\to\mathit{str}}.y(\lambda z^{\mathit{str}}.z + /\mathsf{v}/)$$

if $\Delta(\mathsf{A}, q) = \langle \mathsf{v}, 0, q_f\rangle$ for some $q_f \in Q_f$ for the last element q of $\vec{q}$.

We then define an ACG $\mathscr{G}^T \in \mathbf{G}_{\text{tree,string}}(2, 1(\text{sr}), 4)$ as

$$\mathscr{G}^T = \langle \Sigma_0, \Sigma_1, \Sigma_V, \mathscr{L}_1, \mathscr{L}_2, s\rangle.$$

Lemma 5.26. $\mathcal{O}(\mathscr{G}^T) = \mathcal{R}(T)$.

Proof. $[\mathcal{O}(\mathscr{G}^T) \subseteq \mathcal{R}(T)]$

First we show the following claim by induction on M:

> Suppose that $M \in \mathbb{T}(\Sigma_0)$ has type $[\mathsf{A},\vec{p}] \in \mathscr{A}_0 - \{s\}$. Let $\vec{p}$ be partitioned into k unitary histories $\vec{p}_1, \ldots, \vec{p}_k$ of A. Then, there are $\mathsf{w}_1, \ldots, \mathsf{w}_k \in V^*$ such that
>
> - $\mathscr{L}_2(M) = \lambda x^{\mathit{str}^k\to\mathit{str}}.x/\mathsf{w}_1/\ldots/\mathsf{w}_k/$,
> - $\langle \mathscr{L}_1(M), \varepsilon, p_{i,1}, \varepsilon\rangle \vdash_T^* \langle \mathscr{L}_1(M), \varepsilon, p_{i,k_i+1}, \mathsf{w}_i\rangle$ where $\vec{p}_i = p_{i,1} \ldots p_{i,k_i+1}$.

Basis. Suppose that $M = [\![\mathsf{A}, \langle \mathsf{A}, p_1\rangle \ldots \langle \mathsf{A}, p_k\rangle]\!] \in \mathscr{C}_0$ with $\mathsf{A} \in \Sigma_1^{(0)}$. Every p_i is an up-state and thus a unitary history of A. By the definition,

- $\mathscr{L}_2([\![\mathsf{A}, \langle \mathsf{A}, p_1\rangle \ldots \langle \mathsf{A}, p_k\rangle]\!]) = \lambda x^{\mathit{str}^k\to\mathit{str}}.x\overbrace{/\varepsilon/\ldots/\varepsilon/}^{k \text{ times}}$,
- $\langle \mathsf{A}, \varepsilon, p_i, \varepsilon\rangle \vdash_T^0 \langle \mathsf{A}, \varepsilon, p_i, \varepsilon\rangle$.

Step. Suppose that M is of the form $[\![\mathsf{A}\mathsf{B}_1 \dots \mathsf{B}_n, \sigma]\!] M_1 \dots M_n$ and

$$\tau_0([\![\mathsf{A}\mathsf{B}_1 \dots \mathsf{B}_n, \sigma]\!]) = [\mathsf{B}_1, \vec{q}_1] \to \dots \to [\mathsf{B}_n, \vec{q}_n] \to [\mathsf{A}, \vec{p}].$$

σ must be of the form

$$\begin{aligned} \sigma &= \sigma_1 \dots \sigma_k \\ \sigma_i &= \sigma_{i,1} \dots \sigma_{i,k_i} \langle \mathsf{A}, p_{i,k_i+1} \rangle \\ \sigma_{i,j} &= \langle \mathsf{A}, p_{i,j} \rangle \langle \mathsf{B}_{d_{i,j}}, q_{i,j,1} \rangle \dots \langle \mathsf{B}_{d_{i,j}}, q_{i,j,k_{i,j}+1} \rangle \end{aligned}$$

such that

- the sequence $\vec{p} = p_{1,1} \dots p_{1,k_1+1} \dots p_{k,1} \dots p_{k,k_k+1}$ is a valid history of A, where each $p_{i,1} \dots p_{i,k_i+1}$ is a unitary history of A,
- $\vec{q}_h = \langle\, q_{i,j,g} \mid d_{i,j} = h,\ 1 \le g \le k_{i,j} + 1 \,\rangle$ is a valid history of B_h, where each $q_{i,j,1} \dots q_{i,j,k_{i,j}+1}$ is a unitary history of B_h,

and moreover, for each $1 \le i \le k$ and $1 \le j \le k_i$,

$$\Delta(\mathsf{A}, p_{i,j}) = \langle \mathsf{v}_{i,j}, d_{i,j}, q_{i,j,1} \rangle \text{ with } 1 \le d_{i,j} \le n, \tag{5.3}$$
$$\Delta(\mathsf{B}_{d_{i,j}}, q_{i,j,k_{i,j}+1}) = \langle \mathsf{v}'_{i,j}, 0, p_{i,j+1} \rangle. \tag{5.4}$$

We apply the induction hypothesis to M_h of type $[\mathsf{B}_h, \vec{q}_h]$ for each $1 \le h \le n$. If $\vec{q}_h$ is partitioned into l_h unitary histories of B_h, then there are strings $\mathsf{u}_{h,1}, \dots, \mathsf{u}_{h,l_h} \in V^*$ such that

$$\mathscr{L}_2(M_h) = \lambda x^{\mathit{str}^{l_h} \to \mathit{str}}.x/\mathsf{u}_{h,1}/\dots/\mathsf{u}_{h,l_h}/, \tag{5.5}$$

and for each i, j with $d_{i,j} = h$,

$$\langle \mathscr{L}_1(M_h), \varepsilon, q_{i,j,1}, \varepsilon \rangle \vdash_T^* \langle \mathscr{L}_1(M_h), \varepsilon, q_{i,j,k_{i,j}+1}, \mathsf{w}_{i,j} \rangle \tag{5.6}$$

where $\mathsf{w}_{i,j} = \mathsf{u}_{h,g}$ for the g-th pair $\langle i, j \rangle$ such that $d_{i,j} = h$, i.e., $\langle \mathsf{u}_{h,1}, \dots, \mathsf{u}_{h,l_h} \rangle = \langle\, \mathsf{w}_{i,j} \mid d_{i,j} = h \,\rangle$. Thus, for each $j \le k_i$ we get

$$\begin{aligned} \langle \mathscr{L}_1(M), \varepsilon, p_{i,j}, \varepsilon \rangle &\vdash_T \langle \mathscr{L}_1(M), d_{i,j}, q_{i,j,1}, \mathsf{v}_{i,j} \rangle && \text{(by (5.3))} \\ &\vdash_T^* \langle \mathscr{L}_1(M), d_{i,j}, q_{i,j,k_{i,j}+1}, \mathsf{v}_{i,j}\mathsf{w}_{i,j} \rangle && \text{(by (5.6))} \\ &\vdash_T \langle \mathscr{L}_1(M), \varepsilon, p_{i,j+1}, \mathsf{v}_{i,j}\mathsf{w}_{i,j}\mathsf{v}'_{i,j} \rangle, && \text{(by (5.4))} \end{aligned}$$

and therefore

$$\langle \mathscr{L}_1(M), \varepsilon, p_{i,1}, \varepsilon \rangle \vdash_T^* \langle \mathscr{L}_1(M), \varepsilon, p_{i,k_i+1}, \mathsf{w}_i \rangle$$

for $\mathsf{w}_i = \mathsf{v}_{i,1}\mathsf{w}_{i,1}\mathsf{v}'_{i,1}\dots\mathsf{v}_{i,k_i}\mathsf{w}_{i,k_i}\mathsf{v}'_{i,k_i}$. On the other hand, by the definition of $\mathscr{L}_2$,

$$\mathscr{L}_2([\![\mathsf{A}\mathsf{B}_1\dots\mathsf{B}_n,\sigma]\!]) = \lambda y_1\dots y_n x.y_1(\lambda\vec{z}_1.y_2(\lambda\vec{z}_2.\dots y_n(\lambda\vec{z}_n.xN_1\dots N_k)\dots))$$

where $\vec{z}_h = \langle\, z_{i,j} \mid d_{i,j} = h \,\rangle$ and

$$N_i = /\mathsf{v}_{i,1}/ + z_{i,1} + /\mathsf{v}'_{i,1}/ + \dots + /\mathsf{v}_{i,k_i}/ + z_{i,k_i} + /\mathsf{v}'_{i,k_i}/.$$

Let $/\vec{\mathsf{u}}_h/ = \langle /\mathsf{u}_{h,1}/,\dots,/\mathsf{u}_{h,l_h}/\rangle = \langle\, /\mathsf{w}_{i,j}/ \mid d_{i,j} = h \,\rangle$. Then,

$$\begin{aligned}\mathscr{L}_2(M) &= \mathscr{L}_2([\![\mathsf{A}\mathsf{B}_1\dots\mathsf{B}_n,\sigma]\!])(\lambda x^{\mathit{str}^{l_1}\to\mathit{str}}.x/\vec{\mathsf{u}_1}/)\dots(\lambda x^{\mathit{str}^{l_n}\to\mathit{str}}.x/\vec{\mathsf{u}_n}/) \qquad \text{(by (5.5))}\\ &= \lambda x.xN_1\dots N_k[/\vec{\mathsf{u}}_1//\vec{z}_1]\dots[/\vec{\mathsf{u}}_n//\vec{z}_n]\\ &= \lambda x.xN_1\dots N_k[/\mathsf{w}_{i,j}//z_{i,j}]_{1\le i\le k}^{1\le j\le k_i}\\ &= \lambda x.x/\mathsf{w}_1/\dots/\mathsf{w}_n/.\end{aligned}$$

We have completed the proof of the claim.

Suppose that $M \in \mathcal{A}(\mathscr{G}^T)$. Then, M has the form $M = [\![\mathsf{A},\vec{q}]\!]M'$ for some M' of type $[\mathsf{A},\vec{q}]$, where $\vec{q}$ is a root history of A of the form $\vec{q} = q_s q_2\dots q_{k+1}$ such that

$$\Delta(\mathsf{A}, q_{k+1}) = \langle \mathsf{v}, 0, q\rangle$$

for some $q \in Q_f$ and $\mathsf{v} \in V^*$. By applying the above claim to M', we get

- $\mathscr{L}_2(M') = \lambda x^{\mathit{str}\to\mathit{str}}.x/\mathsf{w}/$,
- $\langle \mathscr{L}_1(M'), \varepsilon, q_s, \varepsilon\rangle \vdash_T^* \langle \mathscr{L}_1(M'), \varepsilon, q_{k+1}, \mathsf{w}\rangle$.

Combining the above derivation with respect to $\mathscr{L}_1(M')$ and $\Delta(\mathsf{A}, q_{k+1}) = \langle \mathsf{v}, 0, q\rangle$, we get

$$\langle \mathscr{L}_1(M'), \varepsilon, q_s, \varepsilon\rangle \vdash_T^* \langle \mathscr{L}_1(M'), \uparrow, q, \mathsf{w}\mathsf{v}\rangle,$$

and thus we have $\langle \mathscr{L}_1(M'), \mathsf{w}\mathsf{v}\rangle \in \mathcal{R}(T)$. Since $\mathscr{L}_1([\![\mathsf{A},\vec{q}]\!]) = \lambda x.x$ and $\mathscr{L}_2([\![\mathsf{A},\vec{q}]\!]) = \lambda y.y(\lambda z.z + /\mathsf{v}/)$, we get

$$\begin{aligned}\mathscr{L}_1([\![\mathsf{A},\vec{q}]\!]M') &= \mathscr{L}_1(M')\\ \mathscr{L}_2([\![\mathsf{A},\vec{q}]\!]M') &= /\mathsf{w}\mathsf{v}/.\end{aligned}$$

We obtain $\mathcal{O}(\mathscr{G}^T) \subseteq \mathcal{R}(T)$.

$[\mathcal{R}(T) \subseteq \mathcal{O}(\mathscr{G}^T)]$

Let K' be a subtree of the correctly decorated tree of an accepted tree, $[\mathsf{A}, \vec{p}]$ the label of the root of K', $\vec{p} = \vec{p}_1 \dots \vec{p}_k$, and $\vec{p}_i = p_{i,1} \dots p_{i,k_i+1}$ where each $\vec{p}_i$ is a unitary history of A. Then, for each $1 \leq i \leq k$ we have

$$\langle K, \varepsilon, p_{i,1}, \varepsilon \rangle \vdash_T^* \langle K, \varepsilon, p_{i,k_i+1}, \mathsf{w}_i \rangle$$

where $K \in \Lambda(\Sigma_1)$ is the tree obtained from K' by stripping off the decoration with states. We first prove that for such K, $\vec{p}$, and $\mathsf{w}_1, \dots, \mathsf{w}_k$, there is $M \in \Lambda(\Sigma_0)$ such that

- $\tau_0(M) = [\mathsf{A}, \vec{p}]$,
- $\mathscr{L}_1(M) = K$,
- $\mathscr{L}_2(M) = \lambda x^{\mathit{str}^k \to \mathit{str}}.x/\mathsf{w}_1/\dots/\mathsf{w}_k/$

by induction on K.

Basis. If $K = \mathsf{A} \in \Sigma_1^{(0)}$ and $K' = [\mathsf{A}, \vec{p}]$ where $\vec{p} = p_1 \dots p_k$ is a valid history of A, then we have

$$\langle K, \varepsilon, p_i, \varepsilon \rangle \vdash_T^0 \langle K, \varepsilon, p_i, \varepsilon \rangle$$

for $1 \leq i \leq k$. By the definition, we have an abstract constant

$$\begin{aligned}
&[\![\mathsf{A}, \langle \mathsf{A}, p_1 \rangle \dots \langle \mathsf{A}, p_k \rangle]\!] \in \mathscr{C}_0 \text{ such that} \\
&\tau_0([\![\mathsf{A}, \langle \mathsf{A}, p_1 \rangle \dots \langle \mathsf{A}, p_k \rangle]\!]) = [\mathsf{A}, \vec{p}], \\
&\mathscr{L}_1([\![\mathsf{A}, \langle \mathsf{A}, p_1 \rangle \dots \langle \mathsf{A}, p_k \rangle]\!]) = \mathsf{A} = K, \\
&\mathscr{L}_2([\![\mathsf{A}, \langle \mathsf{A}, p_1 \rangle \dots \langle \mathsf{A}, p_k \rangle]\!]) = \lambda x^{\mathit{str}^k \to \mathit{str}}.x \underbrace{/\varepsilon/\dots/\varepsilon/}_{k \text{ times}}.
\end{aligned}$$

Step. Suppose that a tree $K \in \mathbb{T}(\Sigma_1)$ and a sequence $\vec{p}$ of states are such that $K = \mathsf{A}(\mathsf{B}_1\vec{K}_1)\dots(\mathsf{B}_n\vec{K}_n)$, $\vec{p} = p_{1,1} \dots p_{1,k_1+1} \dots p_{k,1} \dots p_{k,k_k+1}$ where each $p_{i,1} \dots p_{i,k_i+1}$ is a unitary history of A, and

$$\langle K, \varepsilon, p_{i,1}, \varepsilon \rangle \vdash_T^* \langle K, \varepsilon, p_{i,k_i+1}, \mathsf{w}_i \rangle \tag{5.7}$$

for each $1 \leq i \leq k$.

For each $j \leq k_i$, since $p_{i,j}$ is a down-state for A, we have $\Delta(\mathsf{A}, p_{i,j}) = \langle \mathsf{v}_{i,j}, d_{i,j}, r_{i,j} \rangle$ for some $\mathsf{v}_{i,j} \in V^*$, $d_{i,j} > 0$, $r_{i,j} \in Q$, and T goes down to the $d_{i,j}$-th child of A.

$$\langle K, \varepsilon, p_{i,j}, \varepsilon \rangle \vdash_T \langle K, d_{i,j}, r_{i,j}, \mathsf{v}_{i,j} \rangle \tag{5.8}$$

Since T must come back from $\mathsf{B}_{d_{i,j}}$ to the node A with the state $p_{i,j+1}$, there are $r'_{i,j} \in Q$ and $\mathsf{v}'_{i,j} \in V^*$ such that $\Delta(\mathsf{B}_{d_{i,j}}, r'_{i,j}) = \langle \mathsf{v}'_{i,j}, 0, p_{i,j+1} \rangle$, i.e.,

$$\langle K, d_{i,j}, r'_{i,j}, \varepsilon \rangle \vdash_T \langle K, \varepsilon, p_{i,j+1}, \mathsf{v}'_{i,j} \rangle \tag{5.9}$$

Consider the move of T from when it is at $\mathsf{B}_{d_{i,j}}$ with the state $r_{i,j}$ to when it is at $\mathsf{B}_{d_{i,j}}$ with the state $r'_{i,j}$. That is,

$$\langle K, d_{i,j}, r_{i,j}, \varepsilon\rangle \vdash_T^* \langle K, d_{i,j}, r'_{i,j}, \mathsf{w}_{i,j}\rangle \tag{5.10}$$

for some $\mathsf{w}_{i,j} \in V^*$. By (5.7)–(5.10), we get the equation

$$\mathsf{w}_i = \mathsf{v}_{i,1}\mathsf{w}_{i,1}\mathsf{v}'_{i,1} \dots \mathsf{v}_{i,k_i}\mathsf{w}_{i,k_i}\mathsf{v}'_{i,k_i}. \tag{5.11}$$

Suppose that the h-th child of A is decorated as $[\mathsf{B}_h, \vec{q}_h]$ in K'. $\vec{q}_{d_{i,j}}$ must contain as a subsequence a unitary history of $\mathsf{B}_{d_{i,j}}$ corresponding to (5.10). Let the unitary history be $\vec{q}_{i,j} = q_{i,j,1} \dots q_{i,j,k_{i,j}+1}$, where $q_{i,j,1} = r_{i,j}$, $q_{i,j,k_{i,j}+1} = r'_{i,j}$. We see $\vec{q}_h = \langle\, q_{i,j,g} \mid d_{i,j} = h,\ 1 \le g \le k_{i,j}+1 \,\rangle$. By applying the induction hypothesis to $\mathsf{B}_h\vec{K}_h$ and $\vec{q}_h$, we get $M_h \in \Lambda(\Sigma_0)$ such that

$$\begin{aligned}
\tau_0(M_h) &= [\mathsf{B}_h, \vec{q}_h],\\
\mathscr{L}_1(M_h) &= \mathsf{B}_h\vec{K}_h,\\
\mathscr{L}_2(M_h) &= \lambda x.x\langle /\mathsf{w}_{i,j}/\rangle_{d_{i,j}=h}.
\end{aligned} \tag{5.12}$$

Let σ be defined as

$$\begin{aligned}
\sigma &= \sigma_1 \dots \sigma_k\\
\sigma_i &= \sigma_{i,1} \dots \sigma_{i,k_i}\langle \mathsf{A}, p_{i,k_i+1}\rangle\\
\sigma_{i,j} &= \langle \mathsf{A}, p_{i,j}\rangle\langle \mathsf{B}_{d_{i,j}}, q_{i,j,1}\rangle \dots \langle \mathsf{B}_{d_{i,j}}, q_{i,j,k_{i,j}+1}\rangle.
\end{aligned}$$

Since σ is a locally consistent combination of valid histories of $\langle \mathsf{A}, \mathsf{B}_1, \dots, \mathsf{B}_n\rangle$, we have an abstract constant $[\![\mathsf{A}\mathsf{B}_1 \dots \mathsf{B}_n, \sigma]\!]$ such that

$$\begin{aligned}
\tau_0([\![\mathsf{A}\mathsf{B}_1 \dots \mathsf{B}_n, \sigma]\!]) &= [\mathsf{B}_1, \vec{q}_1] \to \dots \to [\mathsf{B}_n, \vec{q}_n] \to [\mathsf{A}, \vec{p}]\\
\mathscr{L}_1([\![\mathsf{A}\mathsf{B}_1 \dots \mathsf{B}_n, \sigma]\!]) &= \mathsf{A}\\
\mathscr{L}_2([\![\mathsf{A}\mathsf{B}_1 \dots \mathsf{B}_n, \sigma]\!]) &= \lambda y_1 \dots y_n x.y_1(\lambda \vec{z}_1. \dots y_n(\lambda \vec{z}_n.xN_1 \dots N_k) \dots)\\
&\qquad \text{where } N_i = /\mathsf{v}_{i,1}/ + z_{i,1} + /\mathsf{v}'_{i,1}/ + \dots + /\mathsf{v}_{i,k_i}/ + z_{i,k_i} + /\mathsf{v}'_{i,k_i}/\\
&\qquad\quad \text{and } \vec{z}_h = \langle\, z_{i,j} \mid d_{i,j} = h \,\rangle.
\end{aligned}$$

Thus, for the abstract term $M = [\![\mathsf{A}\mathsf{B}_1 \dots \mathsf{B}_n, \sigma]\!]M_1 \dots M_n \in \Lambda(\Sigma_0)$, we have

$$\begin{aligned}
\tau_0(M) &= [\mathsf{A}, \vec{p}]\\
\mathscr{L}_1(M) &= \mathsf{A}(\mathsf{B}_1\vec{K}_1) \dots (\mathsf{B}_n\vec{K}_n)\\
\mathscr{L}_2(M) &= \lambda x.\mathscr{L}_2(M_1)(\lambda \vec{z}_1. \dots \mathscr{L}_2(M_n)(\lambda \vec{z}_n.xN_1 \dots N_k) \dots)\\
&= \lambda x.xN_1 \dots N_k[/\mathsf{w}_{i,j}//z_{i,j}]_{1\le j\le k_i}^{1\le i\le k} && \text{(by (5.12))}\\
&= \lambda x.x/\mathsf{w}_1/ \dots /\mathsf{w}_k/. && \text{(by (5.11))}
\end{aligned}$$

Suppose that $\langle K, w\rangle \in \mathcal{R}(T)$ for some tree $K \in \mathbb{T}(\Sigma_1)$ and $\mathsf{w} \in V^*$. Let $\vec{q} = q_s q_2 \dots q_{k+1}$ be the root history of the root label A of K obtained from the derivation

$$\langle K, \varepsilon, q_s, \varepsilon\rangle \vdash_T^* \langle K, \varepsilon, q_{k+1}, \mathsf{w}'\rangle \vdash_T \langle K, \uparrow, q, \mathsf{w}'\mathsf{v}\rangle$$

where $q \in Q_f$ and $\mathsf{w} = \mathsf{w}'\mathsf{v}$. By applying the above claim to the derivation

$$\langle K, \varepsilon, q_s, \varepsilon\rangle \vdash_T^* \langle K, \varepsilon, q_{k+1}, \mathsf{w}'\rangle,$$

we get a λ-term M such that

- $\tau_0(M) = [\mathsf{A}, \vec{q}]$,
- $\mathscr{L}_1(M) = K$,
- $\mathscr{L}_2(M) = \lambda x.x/\mathsf{w}'/$.

Recall that we have the abstract constant $[\![\mathsf{A}, \vec{q}]\!]$ such that

- $\tau_0([\![\mathsf{A}, \vec{q}]\!]) = [\mathsf{A}, \vec{q}] \to s$,
- $\mathscr{L}_1([\![\mathsf{A}, \vec{q}]\!]) = \lambda x.x$,
- $\mathscr{L}_2([\![\mathsf{A}, \vec{q}]\!]) = \lambda y.y(\lambda z.z + /\mathsf{v}/)$.

Thus, for $[\![\mathsf{A}, \vec{q}]\!]M \in \mathcal{A}(\mathscr{G}^T)$, we get $\mathscr{L}_1([\![\mathsf{A}, \vec{q}]\!]M) = K$ and $\mathscr{L}_2([\![\mathsf{A}, \vec{q}]\!]M) = /\mathsf{w}'\mathsf{v}/ = /\mathsf{w}/$. This completes the proof. □

Proposition 5.27. *For every deterministic tree walking transducer T, there is a 2D-ACG $\mathscr{G}^T \in \mathbf{G}_{\mathrm{tree,string}}(2, 1(\mathrm{r}), 4)$ such that $\mathcal{O}(\mathscr{G}^T) = \mathcal{R}(T)$.*

Proof. The lexicon $\mathscr{L}_1$ defined above is a semi-relabeling. Abstract constants representing root histories are mapped to the identity: $\mathscr{L}_1([\![\mathsf{A}, \vec{q}]\!]) = \lambda x^o.x$. These constants can be eliminated as we have done in Section 4.3.3 to eliminate unary nonlexical constants from second-order ACGs. For each constant $[\![\mathsf{A}\vec{\mathsf{B}}, \sigma]\!]$ of type $\vec{\alpha} \to [\mathsf{A}, \vec{q}]$, we add a new constant of type $\vec{\alpha} \to s$ that is mapped to $\mathscr{L}_i(\lambda\vec{x}.[\![\mathsf{A}, \vec{q}]\!]([\![\mathsf{A}\vec{\mathsf{B}}, \sigma]\!]\vec{x}))$ by the lexicon $\mathscr{L}_i$. Then we can get rid of $[\![\mathsf{A}, \vec{q}]\!]$ from the grammar. □

Remark 5.28. In the original definition [1], a DTWT has a context-free grammar as its parameter and takes only derivation trees of the CFG as input. The above proposition still holds if input is restricted to elements of a regular tree language $\mathcal{L}$. Let $\mathscr{G}^{\mathcal{L}} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s\rangle \in \mathbf{G}_{\mathrm{tree}}(2, 1(\mathrm{r}))$ represent the regular tree language $\mathcal{L}$ and $\mathscr{G}^T \in \mathbf{G}_{\mathrm{tree,string}}(2, 1(\mathrm{r}), 4)$ encoding the tree transducer whose input is not restricted. Applying Lemma 5.21 to the 2D-ACGs $\mathscr{G}^{\mathcal{L}\mathcal{L}} = \langle \Sigma_0, \Sigma_1, \Sigma_1, \mathscr{L}, \mathscr{L}, s\rangle$ and $\mathscr{G}^T \in \mathbf{G}_{\mathrm{tree,string}}(2, 1(\mathrm{r}), 4)$, we obtain the desired 2D-ACG.

Now we obtain the two-dimensional version of Salvati's Theorem (Theorem 2.6). It is easy to extend the following result to second-order kD-ACGs for any $k \in \mathbb{N}$.

Corollary 5.29. *For every second-order 2D-ACG* $\mathscr{G} \in \mathbf{G}_{\text{string,string}}(2, n_1, n_2)$*, there is an equivalent 2D-ACG* $\mathscr{G}' \in \mathbf{G}_{\text{string,string}}(2, 4, 4)$*.*

Proof. In the proof of Salvati's Theorem, for a given second-order string 1D-ACG $\mathscr{G}_0 \in \mathbf{G}_{\text{string}}(2, n)$, he constructs a DTWT T_0 as a string language generator whose output language coincides with the object language of $\mathscr{G}_0$, where input trees for T_0 are restricted to elements of $\mathcal{A}(\mathscr{G}_0)$.[3]

Let $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \Sigma_2, \mathscr{L}_1, \mathscr{L}_2, s \rangle \in \mathbf{G}_{\text{string}}(2, n_1, n_2)$. For the i-th projection $\mathscr{G}_i = \langle \Sigma_0, \Sigma_i, \mathscr{L}_i, s \rangle$ of $\mathscr{G}$, let $T_i = \langle \Sigma_0, \Sigma_i, Q_i, q_i, F \rangle$ be the DTWT obtained by Salvati's method whose input is restricted to a regular tree language $\mathcal{A}(\mathscr{G})$. That is,

$$\mathcal{R}(T_i) = \{ \langle M, P_i \rangle \mid M \in \mathcal{A}(\mathscr{G}),\ P_i = \mathscr{L}_i(M) \}.$$

By applying our method to DTWTs T_i, we get 2D-ACGs $\mathscr{G}_i' \in \mathbf{G}_{\text{tree,string}}(2, 1(\text{r}), 4)$ such that

$$\mathcal{O}(\mathscr{G}_i) = \mathcal{R}(T_i).$$

By Lemma 5.21, we get a 2D-ACG $\mathscr{G}' \in \mathbf{G}(2, 4, 4)$ such that

$$\begin{aligned} \mathcal{O}(\mathscr{G}') &= \{ \langle P_1, P_2 \rangle \mid \langle M, P_i \rangle \in \mathcal{O}(\mathscr{G}_i') = \mathcal{R}(T_i) \} \\ &= \{ \langle P_1, P_2 \rangle \mid M \in \mathcal{A}(\mathscr{G}),\ P_i = \mathscr{L}_i(M) \} = \mathcal{O}(\mathscr{G}). \end{aligned}$$ □

The converse of Proposition 5.27 does not hold, since there is a regular tree language that cannot be recognized by any DTWT [23].

5.6 Linearization of Affine Two-Dimensional ACGs

In Chapter 3, we gave two linearization methods for affine 1D-ACG (Theorem 3.16). One of them is valid for 2D-ACGs as well.

Theorem 5.30. *For every affine 2D-dimensional ACG* $\mathscr{G} \in \mathbf{G}^{\text{aff}}(m, n_1, n_2)$*, there is a linear 2D-dimensional ACG* $\mathscr{G}'' \in \mathbf{G}^{\text{lin}}(m, \max\{2, n_1\}, \max\{2, n_2\})$ *such that* $\mathcal{O}(\mathscr{G}'') = \mathcal{O}(\mathscr{G}) \cap (\Lambda^{\text{lin}}(\Sigma_1) \times \Lambda^{\text{lin}}(\Sigma_2))$*.*

[3] Precisely speaking, Salvati gives a slight modification on the given ACG $\mathscr{G}_0$ before constructing a DTWT. Nevertheless, it is not an obstacle to our discussion at all, since the same modification can be done on 2D-ACGs.

Proof. Let $\mathscr{G}_i = \langle \Sigma_0, \Sigma_i, \mathscr{L}_i, s \rangle$ be the i-th projection of the given affine ACG $\mathscr{G}$, and $\mathscr{G}_i'' = \langle \Sigma_{i,0}, \Sigma_i, \mathscr{L}_i'', [s, \mathscr{L}_i(s)] \rangle$ the linearized form of $\mathscr{G}_i$ obtained by our method. Recall that we have a relabeling lexicon $\mathscr{L}_{0,i} : \Sigma_{0,i} \to \Sigma_0$ that satisfies the equation (3.5), i.e.,

$$\{\, M \in \mathcal{A}(\mathscr{G}_i) \mid |\mathscr{L}(M)|_\beta \text{ is linear} \,\} = \{\, \mathscr{L}_{0,i}(N) \mid N \in \mathcal{A}(\mathscr{G}_i'') \,\}.$$

Let $\mathscr{G}_i' = \langle \Sigma_{i,0}, \Sigma_0, \Sigma_i, \mathscr{L}_{0,i}, \mathscr{L}_i'', [s, \mathscr{L}_i(s)] \rangle$ for $i = 1, 2$. By applying Lemma 5.21 to two linear 2D-ACGs $\mathscr{G}_i'$ with $i = 1, 2$, we get a 2D-ACG $\mathscr{G}''$ such that

$$\mathcal{O}(\mathscr{G}'') = \mathcal{O}(\mathscr{G}) \cap (\Lambda^{\mathrm{lin}}(\Sigma_1) \times \Lambda^{\mathrm{lin}}(\Sigma_2)). \qquad \square$$

We easily apply this method to any affine kD-ACGs with $k \in \mathbb{N}$.

On the other hand, Theorem 3.8 does not hold for two-dimensional second-order affine ACGs, as the following counterexample shows. Let $\mathscr{G} \in \mathbf{G}^{\mathrm{aff}}_{\mathrm{tree,tree}}(2, 1, 1)$ consist of the following lexical entries:

$\mathsf{x} \in \mathscr{C}_0$	$\tau_0(\mathsf{x})$	$\mathscr{L}_1(\mathsf{x})$	$\mathscr{L}_2(\mathsf{x})$
A	s	$\#$	$\#$
B	$s \to s \to s$	$\lambda x^o y^o.\#$	$\lambda x^o y^o.\mathsf{c}xy$

Then,

$$\mathcal{O}(\mathscr{G}) = \{\#\} \times \mathbb{T}(\Sigma_2).$$

Suppose that there is a linear ACG $\mathscr{G}' \in \mathbf{G}^{\mathrm{lin}}_{\mathrm{tree,tree}}(2, 1, 1)$ generating this language. Without loss of generality, we assume that each vocabulary contains neither useless constants nor useless atomic types. For $\mathscr{G}' = \langle \Sigma_0', \Sigma_1, \Sigma_2, \mathscr{L}_1', \mathscr{L}_2', s' \rangle$, let $\mathscr{G}_1' = \langle \Sigma_0', \Sigma_1, \mathscr{L}_1', s' \rangle$. By $\mathcal{O}(\mathscr{G}_1') = \{\#\}$ and thus $\mathscr{C}_1' = \{\#\}$, we easily see that every abstract constant $\mathsf{a} \in \mathscr{C}_0'$ has nullary or unary type, because the only closed linear terms in $\Lambda^{\mathrm{lin}}(\Sigma_1')$ of second-order types are $\lambda x^o.x$ and $\#$. Let

$$\begin{aligned} h = \max \big(&\{\, \mathrm{height}(|\mathscr{L}_2'(\mathsf{a})|_\beta) \mid \mathsf{a} \in \mathscr{C}_0' \text{ and } \tau_0'(\mathsf{a}) = p \in \mathscr{A}_0' \,\} \\ &\cup \{\, \mathrm{height}(|\mathscr{L}_2'(\mathsf{a})\#|_\beta) \mid \mathsf{a} \in \mathscr{C}_0' \text{ and } \tau_0'(\mathsf{a}) = p \to q \,\} \big) \end{aligned}$$

where the *height* of a tree is defined as usual. It is easy to see that $\mathsf{c}PP \in \mathbb{T}(\Sigma_2) - \mathcal{O}_2(\mathscr{G}')$ for any tree $P \in \mathbb{T}(\Sigma_2)$ of height h.

5.7 Summary

Although symmetric treatment of syntax and semantics of natural language is an important appeal of the ACG formalism as de Groote [15] states in his first paper on the ACG, the mathematical properties of two-dimensional extensions of ACGs have not been studied well so far. This chapter has investigated the mathematical properties of 2D-ACGs, particularly the generative capacity of second-order 2D-ACGs. We have evaluated the generative capacity of 2D-ACGs by encoding existing two-dimensional formalisms, namely, linear macro tree transducers (linear MTTs), synchronous tree adjoining grammars and deterministic tree walking transducers.

Table 5.1 summarizes the generative capacity of second-order 2D-ACGs revealed in this chapter and preceding research, which has already shown that finite state transducers [15] and synchronous tree substitution grammars [58] are encodable by 2D-ACGs. The fact that several transducers, which define relations of two languages in asymmetric ways, are encoded by ACGs, which define relations of two languages in symmetric ways, can be thought of as a variation of Nivat's characterization of finite transducers by bimorphisms. Through simulation of transducers, which model syntax-directed semantics, by 2D-ACGs, we have shown how rich the expressive power of 2D-ACGs is.

The encoding methods for linear tree transducers, linear MTTs, and synchronous tree substitution grammars are indeed straightforward as well as de Groote and Pogodalla's encodings for the variety of context-free formalisms. Therefore, ACGs generalize two-dimensional formalisms as well as one-dimensional ones.

There are several subclasses of 2D-ACGs whose generative capacity remains unclear. It is future work to characterize the expressive power of those subclasses.

Table 5.1: Hierarchy of second-order two-dimensional ACGs

Finite State Transducers	$\subseteq$	$\mathbf{G}_{\text{string,string}}(2,2,2)$
Linear Tree Transducers	$=$	$\mathbf{G}_{\text{tree,tree}}(2,1(\text{sr}),1)$
ε-free Linear Tree Transducers	$=$	$\mathbf{G}_{\text{tree,tree}}(2,1(\text{r}),1)$
Linear Macro Tree Transducers	$=$	$\mathbf{G}_{\text{tree,tree}}(2,1(\text{sr}),2)$
ε-free Linear Macro Tree Transducers	$=$	$\mathbf{G}_{\text{tree,tree}}(2,1(\text{r}),2)$
Synchronous Tree Substitution Grammars	$\subseteq$	$\mathbf{G}_{\text{tree,tree}}(2,1,1)$
Synchronous Tree Adjoining Grammars	$\subseteq$	$\mathbf{G}_{\text{tree,tree}}(2,2(\text{mon}),2(\text{mon}))$
Deterministic Tree Walking Transducers	$\subseteq$	$\mathbf{G}_{\text{tree,string}}(2,1(\text{r}),4)$
$\mathbf{G}_{\text{string,string}}(2,n_1,n_2)$ for $n_1,n_2 \geq 4$	$=$	$\mathbf{G}_{\text{string,string}}(2,4,4)$

Chapter 6

Higher-Order Interpolation in the Linear Lambda Calculus

The main result of this chapter has been published as [59].

6.1 Introduction

Parsing with ACGs and Interpolation

As pointed out by de Groote [15] and Pogodalla [42], parsing with ACGs is closely related to *higher-order interpolation* in the linear lambda calculus, though they are still different problems. Let M be a member of the abstract language $\mathcal{A}(\mathscr{G})$ of an ACG $\mathscr{G}$, and M_0 be obtained by replacing all occurrences of constants by fresh variables $x_1, \ldots, x_n$. Then, M can be represented as

$$M = (\lambda x_1 \ldots x_n.M_0)\mathsf{a}_1 \ldots \mathsf{a}_n \in \mathcal{A}(\mathscr{G}) \tag{6.1}$$

for constants $\mathsf{a}_1, \ldots, \mathsf{a}_n$ appearing in M (multiple occurrences of the same constant are allowed). By the lexicon $\mathscr{L}$ of $\mathscr{G}$, the equation (6.1) becomes

$$\mathscr{L}(M) = P = P_0\mathscr{L}(\mathsf{a}_1) \ldots \mathscr{L}(\mathsf{a}_n) \in \mathcal{O}(\mathscr{G}) \tag{6.2}$$

where P_0 is identical to $\mathscr{L}(\lambda x_1 \ldots x_n.M_0)$ except for the type assignment. The problem of checking whether $P \in \Lambda(\Sigma_1)$ belongs to $\mathcal{O}(\mathscr{G})$ thus becomes that of finding a linear combinator $\lambda x_1 \ldots x_n.M_0$ and constants $\mathsf{a}_1, \ldots, \mathsf{a}_n$ satisfying (6.2). If each a_i and thus $P_i = \mathscr{L}(\mathsf{a}_i)$ is known, it reduces to finding a linear combinator X such that

$$P = XP_1 \ldots P_n.$$

This problem is called *higher-order interpolation in the linear lambda calculus*, which is a restricted form of higher-order matching. For instance, if a string ACG has no lexical ambiguity, that is, for each object constant c, there is a unique abstract constant a such that $\mathscr{L}(\mathsf{a})$ contains c, then we can uniquely determine the constants $\mathsf{a}_1, \ldots, \mathsf{a}_n$ that should appear in $M \in \mathcal{A}(\mathscr{G})$, immediately from the given object term P. De Groote [14] has shown the matching problem in the linear lambda calculus is in NP, while it was open in the general case when he introduced ACGs. This gives an explanation why de Groote [15] defines the ACG formalism on *linear* lambda calculus. Motivated by this perspective, this chapter is devoted to investigating the complexity of the higher-order interpolation problem in the linear lambda calculus. We discuss the problem beyond the application of parsing with ACGs.

Higher-Order Matching in the Linear Lambda Calculus

While the second-order unification problem modulo β and $\beta\eta$ [12] and the sixth-order matching problem modulo β are undecidable [32], Stirling [54] recently shows that matching problem modulo $\beta\eta$ is decidable. On the other hand, there are some results on the complexity of matching in the linear lambda calculus. These results hold for both β and $\beta\eta$-matching. De Groote [14] shows that matching in the linear lambda calculus is NP-complete, and by extending his result, Dougherty and Wierzbicki [5] show that matching in the affine lambda calculus is NP-complete. We call matching *linear* if the linearity is imposed on occurrences of unknowns, i.e., each unknown occurs exactly once in the problem instances. Salvati and de Groote [46] prove that NP-hardness of the second-order linear matching problem in the linear lambda calculus. Besides, Salvati [44] give an algorithm for solving linear matching in the linear lambda calculus. There is another result on matching involving the linearity given by Levy [30]. He gives some conditions under which second-order unification with the restriction that a solution must substitute linear λ-terms for unknowns is decidable.

In Salvati and de Groote's paper [46], they also discuss the interpolation problem. Regrettably, Salvati and de Groote's proof of NP-completeness of third-order interpolation in the linear lambda calculus contains an error. In this chapter we correct the flaw and prove NP-completeness of third-order interpolation in the linear lambda calculus.

While no free variable occurs twice or more in a linear λ-term, the number of occurrences of constants is not constrained. Since constants behave like free variables, it is natural to asks whether NP-hardness still holds when we exclude constants from problem instances. This chapter shows that fourth-

order interpolation in the linear lambda calculus is NP-complete even in the absence of constants. Therefore, multiple occurrences of constants do not play an essential role for NP-hardness of higher-order matching in the linear lambda calculus.

6.2 NP-Complete Problems

Definition 6.1. Let $\mathcal{V}$ be a finite set of *Boolean variables*. Introducing a twin $\neg\mathcal{V} = \{\neg v \mid v \in \mathcal{V}\}$ of $\mathcal{V}$, we call elements of $\mathcal{V}$ and $\neg\mathcal{V}$ *positive literals* and *negative literals* respectively. We then define the set of *literals* as $\mathcal{V} \cup \neg\mathcal{V}$. A *formula in conjunctive normal form (CNF)* $\mathcal{F}$ on $\mathcal{V}$ is a collection of *clauses* which are non-empty subsets of $\mathcal{V} \cup \neg\mathcal{V}$. A valuation ψ on $\mathcal{V}$ is a mapping from $\mathcal{V} \cup \neg\mathcal{V}$ to $\{0, 1\}$ such that $\psi(v) + \psi(\neg v) = 1$ for all $v \in \mathcal{V}$. A clause $\mathcal{C} \in \mathcal{F}$ is *satisfied by a valuation ψ via a literal $w \in \mathcal{V} \cup \neg\mathcal{V}$* iff $w \in \mathcal{C}$ and $\psi(w) = 1$. A CNF $\mathcal{F}$ is *satisfied by a valuation ψ* iff every $\mathcal{C} \in \mathcal{F}$ is satisfied by ψ. A CNF $\mathcal{F}$ is *satisfiable* iff there is ψ that satisfies $\mathcal{F}$.

The *satisfiability problem* is the problem of deciding whether a given CNF $\mathcal{F}$ is satisfiable or not. It is well known that the satisfiability problem is NP-complete [4].

Definition 6.2. An *mP-CNF* $\mathcal{F}$ is a CNF such that each positive literal $v \in \mathcal{V}$ occurs exactly m times in $\mathcal{F}$. An *nN-CNF* $\mathcal{F}$ is a CNF such that each negative literal $\neg v \in \neg\mathcal{V}$ occurs exactly n times in $\mathcal{F}$.

A CNF $\mathcal{F}$ is *polarized* iff each clause $\mathcal{C} \in \mathcal{F}$ contains only positive literals or only negative literals. We call a clause $\mathcal{C}$ *positive* if $\mathcal{C} \subseteq \mathcal{V}$, and *negative* if $\mathcal{C} \subseteq \neg\mathcal{V}$.

By combining the above definitions we define an mPnN-CNF to be a CNF which is at the same time an mP-CNF and an nN-CNF.

We call variants of the satisfiability problem whose instances are restricted to CNFs in a special form with the modifier expressing that form.

Theorem 6.3. *The polarized 2P1N-satisfiability problem is NP-complete.*

Proof. For a given CNF $\mathcal{F}$ on $\mathcal{V}$, we may assume that for each $v \in \mathcal{V}$, the positive literal v occurs exactly the same number of times as the negative literal $\neg v$ in $\mathcal{F}$. Otherwise, if v occurs m times more than $\neg v$, we add appropriate number of clauses $\{\neg v, u_i, \neg u_i\}$ to $\mathcal{F}$ for $1 \leq i \leq m$ where u_i are new Boolean variables. If v occurs m times more than $\neg v$, add clauses $\{v, u_i, \neg u_i\}$ for $1 \leq i \leq m$. Now, for a given CNF $\mathcal{F}$ on $\mathcal{V} = \{v_1, \ldots, v_l\}$ such that the positive literal v_i occurs exactly the same number of times as the

negative literal $\neg v_i$ for each Boolean variable $v_i \in \mathcal{V}$, we construct a polarized 2P1N-CNF $\mathcal{F}'$ on $\mathcal{V}'$ such that $\mathcal{F}$ is satisfiable iff $\mathcal{F}'$ is satisfiable. Let m_i be the number of occurrences of the positive literal v_i (thus of the negative literal $\neg v_i$) for each $v_i \in \mathcal{V}$.

1. Let $\mathcal{V}' = \{\, v_{i,j}, \bar{v}_{i,j} \mid 1 \leq i \leq l,\, 1 \leq j \leq m_i \,\}$.

2. Replace the j-th occurrence of the positive literal v_i in $\mathcal{F}$ with $v_{i,j}$, and the j-th occurrence of the negative literal $\neg v_i$ in $\mathcal{F}$ with $\bar{v}_{i,j}$ for $1 \leq j \leq m_i$.

3. Add the clauses $\{v_{i,j}, \bar{v}_{i,j}\}$ for $1 \leq j \leq m_i$, $\{\neg v_{i,j}, \neg \bar{v}_{i,j+1}\}$ for $1 \leq j < m_i$, and $\{\neg v_{i,m_i}, \neg \bar{v}_{i,1}\}$.

By the construction, $\mathcal{F}'$ is a polarized 2P1N-CNF. It is clear that if $\mathcal{F}$ is satisfied by a valuation ψ, then $\mathcal{F}'$ is also satisfied by ψ' such that $\psi'(v_{i,j}) = \psi(v_i)$ and $\psi'(\bar{v}_{i,j}) = \psi(\neg v_i)$ for all $j \in \{1, \ldots, m_i\}$.

Conversely, suppose that $\mathcal{F}'$ is satisfied by a valuation ψ'. We show that $\psi'(v_{i,j}) = \psi'(v_{i,1})$ and $\psi'(\bar{v}_{i,j}) = \psi'(\neg v_{i,1})$ for all $j \in \{1, \ldots, m_i\}$. Then we easily see that the valuation ψ such that $\psi(v_i) = \psi'(v_{i,1})$ satisfies $\mathcal{F}$.

If $\psi'(v_{i,1}) = 1$, then $\psi'(\bar{v}_{i,2}) = 0$ by $\{\neg v_{i,1}, \neg \bar{v}_{i,2}\} \in \mathcal{F}'$. By $\{v_{i,2}, \bar{v}_{i,2}\} \in \mathcal{F}'$, we see $\psi'(v_{i,2}) = 1$. This way we see that $\psi'(v_{i,j}) = 1$ and $\psi'(\bar{v}_{i,j}) = 0$ for $2 \leq j \leq m_i$. By $\{\neg v_{i,m_i}, \neg \bar{v}_{i,1}\} \in \mathcal{F}'$, we get $\psi'(\bar{v}_{i,1}) = 0$.

If $\psi'(v_{i,1}) = 0$, then $\psi'(\bar{v}_{i,1}) = 1$ by $\{v_{i,1}, \bar{v}_{i,1}\} \in \mathcal{F}'$. By $\{\neg v_{i,m_i}, \neg \bar{v}_{i,1}\} \in \mathcal{F}'$, we see $\psi'(v_{i,m_i}) = 0$. Similarly to the previous case, we see that $\psi'(v_{i,j}) = 0$ and $\psi'(\bar{v}_{i,j}) = 1$ for all $j \in \{1, \ldots, m_i\}$. □

Definition 6.4. The *string rearrangement problem* is the problem of determining whether there is a permutation π on $\{1, \ldots, m\}$ such that $\mathsf{w} = \mathsf{w}_{\pi(1)} \ldots \mathsf{w}_{\pi(m)}$ for a given string w and a sequence of string $\langle \mathsf{w}_1, \ldots, \mathsf{w}_m \rangle$.

Theorem 6.5. *The string rearrangement problem is NP-complete.*

Proof. Clearly the problem is in NP. The NP-hardness is shown by a reduction from the polarized[1] 2P1N-satisfiability problem. Let $\mathcal{F} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m, \mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a polarized 2P1N-CNF on $\mathcal{V} = \{v_1, \ldots, v_l\}$ where each $\mathcal{C}_j$ is a positive clause and each $\mathcal{D}_k$ is a negative clause. Since for any valuation ψ, for each variable $v_i \in \mathcal{V}$, there are at most two clauses that are satisfied by ψ via v_i or $\neg v_i$, we can assume that $2l \geq m + n$. If $2l < m + n$ (it is decidable in polynomial time), $\mathcal{F}$ is not satisfiable.

[1]The polarity is not a mandatory requirement for this proof.

We define strings $\mathsf{w}, \mathsf{x}_i, \mathsf{y}_j$ for $1 \leq i \leq m$, $1 \leq j \leq n$ on the alphabet $V = \{\mathsf{c}_1, \ldots, \mathsf{c}_m, \mathsf{d}_1, \ldots, \mathsf{d}_n, \mathsf{f}\}$ by

$$\begin{aligned}
&\mathsf{w} = \mathsf{v}_1 \ldots \mathsf{v}_l \text{ with } \mathsf{v}_i = \mathsf{f}\mathsf{c}_{\mu(i,1)}\mathsf{f}\mathsf{d}_{\nu(i)}\mathsf{f}\mathsf{c}_{\mu(i,2)}\mathsf{f},\\
&\quad \text{where } \mu(i,h) = j \text{ if the } h\text{-th occurrence of } v_i \text{ is in } \mathcal{C}_j \text{ for } h = 1, 2,\\
&\quad \text{and } \nu(i) = j \text{ if the unique occurrence of } \neg v_i \text{ is in } \mathcal{D}_j,\\
&\mathsf{x}_i = \mathsf{f}\mathsf{c}_i\mathsf{f} \text{ for } 1 \leq i \leq m,\\
&\mathsf{y}_i = \mathsf{f}\mathsf{d}_i\mathsf{f} \text{ for } 1 \leq i \leq n.
\end{aligned}$$

We see that

$$\#_{\mathsf{a}}(\mathsf{w}) \geq \sum_{1 \leq i \leq m} \#_{\mathsf{a}}(\mathsf{x}_i) + \sum_{1 \leq i \leq n} \#_{\mathsf{a}}(\mathsf{y}_i)$$

for all $\mathsf{a} \in V$. If a is either c_j or d_k, then $\sum_{1 \leq i \leq m} \#_{\mathsf{a}}(\mathsf{x}_i) + \sum_{1 \leq i \leq n} \#_{\mathsf{a}}(\mathsf{y}_i) = 1$. Since each clause is not empty, $\#_{\mathsf{a}}(\mathsf{w}) \geq 1$. If $\mathsf{a} = \mathsf{f}$, then $\#_{\mathsf{a}}(\mathsf{w}) = 4l \geq \sum_{1 \leq i \leq m} \#_{\mathsf{a}}(\mathsf{x}_i) + \sum_{1 \leq i \leq n} \#_{\mathsf{a}}(\mathsf{y}_i) = 2(m+n)$ by the assumption that $2l \geq m + n$. Therefore, because the length of w is $7l$ and the sum of the lengths of x_i and y_j for all $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$ is $3(m+n)$, one can find $\mathsf{z}_1, \ldots, \mathsf{z}_{7l-3(m+n)} \in V$ such that

$$\#_{\mathsf{a}}(\mathsf{w}) = \sum_{1 \leq i \leq m} \#_{\mathsf{a}}(\mathsf{x}_i) + \sum_{1 \leq i \leq n} \#_{\mathsf{a}}(\mathsf{y}_i) + \sum_{1 \leq i \leq 7l-3(m+n)} \#_{\mathsf{a}}(\mathsf{z}_i)$$

for all $\mathsf{a} \in V$.

We show that there is a valuation ψ that satisfies $\mathcal{F}$ iff there is a permutation on the sequence $\langle \mathsf{x}_1, \ldots, \mathsf{x}_m, \mathsf{y}_1, \ldots, \mathsf{y}_n, \mathsf{z}_1, \ldots, \mathsf{z}_{7l-3(m+n)} \rangle$ that rearranges it to coincide with w.

First we show the "if" direction. Suppose that $\langle \mathsf{x}_1, \ldots, \mathsf{x}_m, \mathsf{y}_1, \ldots, \mathsf{y}_n, \mathsf{z}_1, \ldots, \mathsf{z}_{7l-3(m+n)} \rangle$ can be reordered to be equivalent to w. Then, each x_j for $1 \leq j \leq m$ is used as a substring of v_i for some i. and each y_j for $1 \leq j \leq n$ is used as a substring of v_i for some i. We define a valuation ψ on $\mathcal{V}$ as follows:

$$\psi(v_i) = \begin{cases} 1 & \text{if } \mathsf{x}_j \text{ is used as a substring of } \mathsf{v}_i \text{ for some } j,\\ 0 & \text{if } \mathsf{y}_k \text{ is used as a substring of } \mathsf{v}_i \text{ for some } k,\\ \text{any value} & \text{otherwise.} \end{cases}$$

Indeed ψ is well-defined. We cannot use both x_j and y_k as substrings of v_i at the same time, because of the occurrences of the symbol f. Each clause $\mathcal{C}_j$ is satisfied by ψ with the literal v_i if x_j is used as a substring of v_i, and $\mathcal{D}_k$ is satisfied by ψ with the literal $\neg v_i$ if y_k is used as a substring of v_i.

Conversely, if ψ satisfies $\mathcal{F}$, one can find $\phi : \mathcal{F} \to \mathcal{V} \cup \neg\mathcal{V}$ such that $\phi(\mathcal{C}) \in \mathcal{C}$ and $\psi(\phi(\mathcal{C})) = 1$ for all $\mathcal{C} \in \{\mathcal{C}_1, \ldots, \mathcal{C}_m, \mathcal{D}_1, \ldots, \mathcal{D}_n\}$. It never happens that $\phi(\mathcal{C}_j) = v_i$ and $\phi(\mathcal{D}_k) = \neg v_i$. Thus we can use each string x_j as a substring of v_i if $\phi(\mathcal{C}_j) = v_i$, and each string y_k as a substring of v_i if $\phi(\mathcal{D}_k) = v_i$. By the definition of z_i for $1 \leq i \leq 7l - 3(m+n)$, we can reorder $\mathsf{x}_1, \ldots, \mathsf{x}_m, \mathsf{y}_1, \ldots, \mathsf{y}_n, \mathsf{z}_1, \ldots, \mathsf{z}_{7l-3(m+n)}$ to be equivalent to w. □

6.3 Definitions

In this chapter, we extend the notion of λ-term introducing new atomic terms, called *unknowns*. In order to distinguish the extended class of λ-terms from usual unknown-free λ-terms, we call the latter *pure terms*. The set of unknowns are denoted by $\mathscr{U}$ and the letters $\mathbf{X}$ and $\mathbf{Y}$ are used for unknowns. Unknowns can be considered as a kind of variables that are never bound, and should be substituted for by pure terms.

The definitions of $\Lambda(\Sigma)$ and the type $\tau(M)$ of a term $M \in \Lambda(\Sigma)$ are now extended as follows:

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

Other notions related to λ-terms, such as closed terms, linear terms, etc., are defined in the same way as in Chapter 2. Note that a closed term can contain unknowns and that unknowns may occur any number of times in a linear term.

Definition 6.6. A *unification equation* is a pair of closed terms $\langle L, R\rangle$ of the same type. Let $\vec{\mathbf{X}}$ be the unknowns in $\langle L, R\rangle$. A substitution $\sigma = [\vec{N}/\vec{\mathbf{X}}]$ where $\vec{N}$ is a sequence of pure terms is a *solution for* $\langle L, R\rangle$ *modulo* β iff $L\sigma =_\beta R\sigma$. σ is a *solution for* $\langle L, R\rangle$ *modulo* $\beta\eta$ iff $L\sigma =_{\beta\eta} R\sigma$. A *matching equation* is a unification equation $\langle L, R\rangle$ such that R is a pure term. An *interpolation equation* is a matching equation $\langle L, R\rangle$ such that L has just one occurrence of just one unknown as its head, i.e., $L \equiv \mathbf{X}L_1 \ldots L_m$, where

each L_i contains no unknowns. The *order* of a unification equation is defined to be the maximum of the orders of the types of the unknowns appearing in the equation. The *unification, matching,* and *interpolation problem modulo β ($\beta\eta$)* are decision problems which ask whether or not there is a solution modulo β ($\beta\eta$) for given unification, matching, interpolation equations respectively. The unification, matching, and interpolation problem *in the linear lambda calculus* allow only linear λ-terms as problem instances and solutions.

6.4 Interpolation in the Linear Lambda Calculus

While every interpolation equation $\langle \mathbf{X}L_1 \dots L_m, R\rangle$ has a trivial solution $[\lambda x_1 \dots x_m.R/\mathbf{X}]$ ($x_i \notin \mathrm{Fv}(R)$) in the (general) lambda calculus, the interpolation problem in the *linear* lambda calculus is not trivial. That the interpolation problem in the linear lambda calculus is in NP is an immediate corollary of the following theorem.

Theorem 6.7 (de Groote [14]). *The matching problem in the linear lambda calculus is NP-complete.*

Theorem 6.8 (Salvati and de Groote [46]). *Second-order matching in the linear lambda calculus is NP-complete even if every unknown appears exactly once.*

Salvati and de Groote [46, Proposition 3] have claimed that *third-order interpolation in the linear lambda calculus is NP-complete* by a reduction from the 1N-satisfiability problem (Theorem 6.3). For a given 1N-CNF $\mathcal{F} = \{\mathcal{C}_1, \dots, \mathcal{C}_m\}$ on $\mathcal{V} = \{v_1, \dots, v_l\}$, they define a third-order interpolation equation $\langle L, R\rangle$ on $\Sigma = \langle \mathscr{A}, \mathscr{C}, \tau\rangle$ as follows:[2]

$$
\begin{aligned}
&\mathscr{A} = \{o\},\ \mathscr{C} = \{\mathsf{a}, \mathsf{c}_j, \mathsf{g} \mid 1 \le j \le m\},\\
&\quad \tau(\mathsf{a}) = o,\ \tau(\mathsf{c}_j) = o^m \to o,\ \tau(\mathsf{g}) = o^l \to o,\\
&L \equiv \mathbf{X}(\lambda x_1^o \dots x_m^o.\mathsf{c}_1 x_1 \dots x_m) \dots (\lambda x_1^o \dots x_m^o.\mathsf{c}_m x_1 \dots x_m),\\
&R \equiv \mathsf{g} V_1 \dots V_l \text{ for } V_i \equiv N_i P_{i,1} \dots P_{i,m}\\
&\qquad \text{where } N_i \equiv \mathsf{c}_j \text{ for } \neg v_i \in \mathcal{C}_j \text{ and } P_{i,j} \equiv \begin{cases} \mathsf{c}_j \mathsf{a}^m & \text{if } v_i \in \mathcal{C}_j,\\ \mathsf{a} & \text{otherwise.}\end{cases}
\end{aligned}
$$

[2] An inessential change is made to the original reduction to facilitate comparison with our reduction.

We present a counterexample to their claim that $\mathcal{F}$ is satisfiable iff $\langle L, R\rangle$ has a solution. Consider their reduction from the following 1N-CNF $\mathcal{F}$:

$$\begin{aligned}&\textsc{Instance}\ \mathcal{F}: \quad \mathcal{C}_1 = \{v_1\},\ \mathcal{C}_2 = \{\neg v_1\}\\ &\textsc{Reduction}\ \langle L, R\rangle: \quad L \equiv \mathbf{X}(\lambda x_1 x_2.\mathsf{c}_1 x_1 x_2)(\lambda x_1 x_2.\mathsf{c}_2 x_1 x_2)\\ &\qquad\qquad\qquad\quad R \equiv \mathsf{g}\big(\mathsf{c}_2(\mathsf{c}_1\mathsf{a}\mathsf{a})\mathsf{a}\big)\end{aligned}$$

If their reduction is correct, $\langle L, R\rangle$ has no solution, since $\mathcal{F}$ is not satisfiable. In fact, however, $[\lambda y_1^{o^2\to o} y_2^{o^2\to o}.\mathsf{g}\big(y_2(y_1\mathsf{a}\mathsf{a})\mathsf{a}\big)/\mathbf{X}]$ is a solution for $\langle L, R\rangle$.

Actually, the proposition that *third-order interpolation in the linear lambda calculus is NP-complete* is an easy corollary to the NP-hardness of the string rearrangement problem.

Proposition 6.9. *Third-order interpolation modulo β ($\beta\eta$) in the linear lambda calculus is NP-complete even if all the constants, variables (other than the unknown) and arguments of the unknown in the problem instances have types o or $o \to o$.*

Proof. Let the pair of w and $\langle \mathsf{w}_1, \ldots, \mathsf{w}_m\rangle$ on an alphabet be an instance of the string rearrangement problem. Clearly it has a solution iff the interpolation equation

$$\langle \mathbf{X}/\mathsf{w}_1/\ldots/\mathsf{w}_m/,\ /\mathsf{w}/\rangle$$

admits a solution. □

It is not hard to show that the second-order interpolation problem in the linear lambda calculus is in P. If a given instance $\langle \mathbf{X}L_1 \ldots L_m, R\rangle$ of the interpolation problem in the linear lambda calculus is second-order, then all L_i have atomic types. $\langle \mathbf{X}L_1 \ldots L_m, R\rangle$ has a solution $[\lambda z_1 \ldots z_m.S/\mathbf{X}]$ iff R can be represented as $R \equiv S[L_1/z_1, \ldots, L_m/z_m]$, because substituting a term of an atomic type for a variable produces no new β-redex. Thus, the following procedure determines whether $\langle \mathbf{X}L_1 \ldots L_m, R\rangle$ admits a solution.

---

let $\mathcal{L} := \{L_1, \ldots, L_m\}$ and $S := R$;
while $\mathcal{L} \neq \varnothing$ **do**
  take $L_i \in \mathcal{L}$ such that L_i is not a proper subterm of any elements of $\mathcal{L}$;
  let $\mathcal{L} := \mathcal{L} - \{L_i\}$;
  if there is S' such that $S = S'[L_i/z_i]$ and z_i occurs exactly once in S'
    then let $S := S'$
    else output "NO" and **halt**;
  end if
end while
output "YES" ($\lambda z_1 \ldots z_m.S$ is a solution);

In the next section, we will show that interpolation in the linear lambda-calculus is still NP-hard even in the absence of constants. For this purpose, we present another proof for the NP-hardness of third-order interpolation in the linear lambda calculus by a reduction from the polarized 1N-satisfiability problem in the remainder of this section. The reduction is an elaboration of the one by Salvati and de Groote. We then eliminate constants from the new reduction with a certain technique, which does not work for the simple reduction in the proof for Proposition 6.9.

Definition 6.10. Let $\mathcal{F} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m, \mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a polarized 1N-CNF on $\mathcal{V} = \{v_1, \ldots, v_l\}$ where each $\mathcal{C}_j$ is a positive clause and each $\mathcal{D}_k$ is a negative clause. First we introduce the following two functions $\mu : \{1, \ldots, l\} \times \{1, \ldots, m\} \to \{0, 1, \ldots, m\}$ and $\nu : \{1, \ldots, l\} \to \{1, \ldots, n\}$ which represent the positive and negative occurrences of each Boolean variable respectively:

$$\mu_{\mathcal{F}}(i, j) = \begin{cases} j & \text{if } v_i \in \mathcal{C}_j, \\ 0 & \text{otherwise,} \end{cases}$$
$$\nu_{\mathcal{F}}(i) = k \text{ for } \neg v_i \in \mathcal{D}_k.$$

We then define an interpolation equation $\langle L_{\mathcal{F}}, R_{\mathcal{F}} \rangle$ on a higher-order signature $\Sigma = \langle \mathscr{A}, \mathscr{C}, \tau \rangle$, where $\mathscr{A} = \{o\}$, $\mathscr{C} = \{\mathsf{c}_j, \mathsf{d}_k, \mathsf{f}, \mathsf{g} \mid 0 \leq j \leq m \text{ and } 1 \leq k \leq n\}$, and $\tau(\mathsf{c}_j) = o,\ \tau(\mathsf{d}_k) = o^m \to o,\ \tau(\mathsf{f}) = o \to o,\ \tau(\mathsf{g}) = o^l \to o$. Let

$$L_{\mathcal{F}} \equiv \mathbf{X} C_1 \ldots C_m D_1 \ldots D_n$$
$$\text{where } \begin{cases} C_j \equiv \mathsf{f}\mathsf{c}_j, \\ D_k \equiv \lambda x_1^o \ldots x_m^o . \mathsf{d}_k(\mathsf{f}x_1) \ldots (\mathsf{f}x_m), \\ \tau(\mathbf{X}) = o^m \to (o^m \to o)^n \to o \text{ and } \mathrm{ord}(\mathbf{X}) = 3, \end{cases}$$
$$R_{\mathcal{F}} \equiv \mathsf{g} V_1 \ldots V_l$$
$$\text{where } V_i \equiv \mathsf{d}_{\nu(i)}(\mathsf{f}\mathsf{c}_{\mu(i,1)}) \ldots (\mathsf{f}\mathsf{c}_{\mu(i,m)}).$$

The intuition behind the reduction is the following. Each C_j in $L_{\mathcal{F}}$ represents the positive clause $\mathcal{C}_j$ and each D_k in $L_{\mathcal{F}}$ represents the negative clause $\mathcal{D}_k$. Each V_i in $R_{\mathcal{F}}$ represents the occurrences of the Boolean variable $v_i \in \mathcal{V}$ in the clauses of $\mathcal{F}$. V_i contains c_j for $j \neq 0$ (*respectively* d_k) iff v_i appears in $\mathcal{C}_j$ (*resp.* $\neg v_i$ appears in $\mathcal{D}_k$). If $\mathcal{F}$ is satisfied by a valuation ψ, then for each $\mathcal{C}_j$ (*resp.* $\mathcal{D}_k$), there is $v_i \in \mathcal{C}_j$ (*resp.* $\neg v_i \in \mathcal{C}_k$) such that $\psi(v_i) = 1$ (*resp.* $\psi(\neg v_i) = 1$). In this case, we can construct a solution $[\lambda y_1 \ldots y_m z_1 \ldots z_n . \mathsf{g} U_1 \ldots U_l / \mathbf{X}]$ which puts the argument C_j (*resp.* D_k) of $\mathbf{X}$ into U_i via y_j (*resp.* z_k) and makes U_i equivalent to V_i by β-reduction (see

Example 6.11). Conversely if $\langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle$ has a solution $[S/\mathbf{X}]$, then S must be of the form $\lambda y_1 \ldots y_m z_1 \ldots z_n.\mathsf{g}U_1 \ldots U_l$ and must put each argument C_j (*resp.* D_k) of $\mathbf{X}$ into U_i for some i via y_j (*resp.* z_k) by the linearity. Then, one can find a valuation ψ such that if S puts the argument C_j (*resp.* D_k) into U_i, then ψ satisfies $\mathcal{C}_j$ via v_i (*resp.* $\mathcal{D}_k$ via $\neg v_i$). The presence of the constant f is the essential difference between Salvati and de Groote's reduction [46] and ours. Due to the number of occurrences of f in V_i for each i, C_j and D_k cannot simultaneously be put into the same U_i for any j and k. This corresponds to the fact that any valuation ψ on $\mathcal{V}$ cannot simultaneously satisfy $\mathcal{C}_j$ via v_i and $\mathcal{D}_k$ via $\neg v_i$ (see Example 6.12).

Example 6.11. INSTANCE $\mathcal{F}$: $\mathcal{C}_1 = \{v_1\},\ \mathcal{C}_2 = \{v_1, v_2\},\ \mathcal{D}_1 = \{\neg v_1, \neg v_2\}$

$$\begin{aligned}\text{REDUCTION } \langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle : \quad & L_{\mathcal{F}} \equiv \mathbf{X}(\mathsf{fc}_1)(\mathsf{fc}_2)\big(\lambda x_1 x_2.\mathsf{d}_1(\mathsf{f}x_1)(\mathsf{f}x_2)\big)\\ & R_{\mathcal{F}} \equiv \mathsf{g}\big(\mathsf{d}_1(\mathsf{fc}_1)(\mathsf{fc}_2)\big)\big(\mathsf{d}_1(\mathsf{fc}_0)(\mathsf{fc}_2)\big)\end{aligned}$$

Let ψ be defined as $\psi(v_1) = 1, \psi(v_2) = 0$. Corresponding to the fact that ψ satisfies $\mathcal{C}_1$ via v_1, $\mathcal{C}_2$ via v_1, and $\mathcal{D}_1$ via $\neg v_2$, we give a solution $[\lambda y_1 y_2 z_1.\mathsf{g}U_1U_2/\mathbf{X}]$ which puts the argument C_1 of $\mathbf{X}$ into U_1, C_2 into U_1, and D_1 into U_2. That is, we give a solution $[S/\mathbf{X}]$ where

$$S \equiv \lambda y_1 y_2 z_1.\mathsf{g}(\mathsf{d}_1 y_1 y_2)(z_1\mathsf{c}_0\mathsf{c}_2).$$

Indeed, we obtain

$$L_{\mathcal{F}}[S/\mathbf{X}] \equiv SC_1C_2D_1 \twoheadrightarrow_\beta \mathsf{g}(\mathsf{d}_1C_1C_2)(D_1\mathsf{c}_0\mathsf{c}_2) \twoheadrightarrow_\beta R_{\mathcal{F}}.$$

Example 6.12. INSTANCE $\mathcal{F}$: $\mathcal{C}_1 = \{v_1\},\ \mathcal{C}_2 = \{v_2\},\ \mathcal{D}_1 = \{\neg v_1, \neg v_2\}$

$$\begin{aligned}\text{REDUCTION } \langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle : \quad & L_{\mathcal{F}} \equiv \mathbf{X}(\mathsf{fc}_1)(\mathsf{fc}_2)\big(\lambda x_1 x_2.\mathsf{d}_1(\mathsf{f}x_1)(\mathsf{f}x_2)\big)\\ & R_{\mathcal{F}} \equiv \mathsf{g}\big(\mathsf{d}_1(\mathsf{fc}_1)(\mathsf{fc}_0)\big)\big(\mathsf{d}_1(\mathsf{fc}_0)(\mathsf{fc}_2)\big)\end{aligned}$$

$\mathcal{F}$ is not satisfiable and $\langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle$ has no solution. e.g.,

$$\begin{aligned}& L_{\mathcal{F}}[\lambda y_1 y_2 z_1.\mathsf{g}\big(\mathsf{d}_1 y_1(\mathsf{fc}_0)\big)\big(z_1\mathsf{c}_0 y_2\big)/\mathbf{X}]\\ & \twoheadrightarrow_\beta \mathsf{g}\big(\mathsf{d}_1(\mathsf{fc}_1)(\mathsf{fc}_0)\big)\big(\mathsf{d}_1(\mathsf{fc}_0)(\mathsf{f}(\mathsf{fc}_2))\big)\\ & \neq_\beta R_{\mathcal{F}}.\end{aligned}$$

Lemma 6.13. *$\langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle$ admits a solution whenever $\mathcal{F}$ is satisfiable.*

Proof. Suppose that $\mathcal{F}$ is satisfied by a valuation ψ. Then, for each clause of $\mathcal{F}$, one can choose a literal via which ψ satisfies the clause. Let a function

ϕ from $\mathcal{F}$ to $\mathcal{V} \cup \neg\mathcal{V}$ be such a choice. That is, $\phi(\mathcal{C}_j) \in \mathcal{C}_j$, $\psi(\phi(\mathcal{C}_j)) = 1$, $\phi(\mathcal{D}_k) \in \mathcal{D}_k$, and $\psi(\phi(\mathcal{D}_k)) = 1$. Define S by

$$S \equiv \lambda y_1 \dots y_m z_1 \dots z_n . \mathsf{g} U_1 \dots U_l$$

$$U_i \equiv \begin{cases} z_{\nu(i)} \mathsf{c}_{\mu(i,1)} \dots \mathsf{c}_{\mu(i,m)} & \text{if } \phi(\mathcal{D}_{\nu(i)}) = \neg v_i, \\ \mathsf{d}_{\nu(i)} U_{i,1} \dots U_{i,m} & \text{otherwise,} \end{cases}$$

$$U_{i,j} \equiv \begin{cases} y_j & \text{if } \phi(\mathcal{C}_j) = v_i, \\ \mathsf{fc}_{\mu(i,j)} & \text{otherwise.} \end{cases}$$

First we confirm the linearity of S. Each z_k indeed occurs exactly once in S, since z_k appears in U_i iff $\phi(\mathcal{D}_k) = \neg v_i$. For y_j, let i be such that $\phi(\mathcal{C}_j) = v_i$. Then, $\psi(v_i) = 1$ and thus $\phi(\mathcal{D}_{\nu(i)}) \neq \neg v_i$ because $\psi(\phi(\mathcal{D}_{\nu(i)})) = 1$. Hence, $U_i \equiv \mathsf{d}_{\nu(i)} U_{i,1} \dots U_{i,m}$ and the only occurrence of y_j in S is in $U_{i,j}$. Therefore, S is a linear λ-term.

In order to see that $L_{\mathcal{F}}[S/\mathbf{X}] \twoheadrightarrow_\beta R_{\mathcal{F}}$, it is enough to check that $U_i\sigma \twoheadrightarrow_\beta V_i$ for the substitution $\sigma = [C_1/y_1, \dots, C_m/y_m,\ D_1/z_1, \dots, D_n/z_n]$. If $\phi(\mathcal{D}_{\nu(i)}) = \neg v_i$, then

$$\begin{aligned} U_i\sigma &\equiv D_{\nu(i)} \mathsf{c}_{\mu(i,1)} \dots \mathsf{c}_{\mu(i,m)} \\ &\equiv (\lambda x_1 \dots x_m . \mathsf{d}_{\nu(i)} (\mathsf{f} x_1) \dots (\mathsf{f} x_m)) \mathsf{c}_{\mu(i,1)} \dots \mathsf{c}_{\mu(i,m)} \\ &\twoheadrightarrow_\beta \mathsf{d}_{\nu(i)} (\mathsf{fc}_{\mu(i,1)}) \dots (\mathsf{fc}_{\mu(i,m)}) \\ &\equiv V_i. \end{aligned}$$

If $\phi(\mathcal{D}_{\nu(i)}) \neq \neg v_i$, then it is easy to see that $U_{i,j}\sigma \equiv \mathsf{fc}_{\mu(i,j)}$. If $\phi(\mathcal{C}_j) \neq v_i$, $U_{i,j}\sigma \equiv \mathsf{fc}_{\mu(i,j)}\sigma \equiv \mathsf{fc}_{\mu(i,j)}$. Otherwise, $\phi(\mathcal{C}_j) = v_i$ implies $\mu(i,j) = j$ and $U_{i,j}\sigma \equiv y_j\sigma \equiv C_j \equiv \mathsf{fc}_j \equiv \mathsf{fc}_{\mu(i,j)}$. Thus,

$$U_i\sigma \equiv \mathsf{d}_{\nu(i)} U_{i,1} \dots U_{i,m}\sigma \twoheadrightarrow_\beta \mathsf{d}_{\nu(i)} (\mathsf{fc}_{\mu(i,1)}) \dots (\mathsf{fc}_{\mu(i,m)}) \equiv V_i. \qquad \square$$

Lemma 6.14. *$\mathcal{F}$ is satisfiable whenever $\langle L_{\mathcal{F}}, R_{\mathcal{F}} \rangle$ admits a solution.*

Proof. Suppose that $[S/\mathbf{X}]$ is a solution for $\langle L_{\mathcal{F}}, R_{\mathcal{F}} \rangle$. We can assume that S is in long normal form and $S \equiv \lambda y_1 \dots y_m z_1 \dots z_n . S'$. Let σ denote the substitution $[\vec{C}/\vec{y},\ \vec{D}/\vec{z}]$. Since $S'\sigma \twoheadrightarrow_\beta R_{\mathcal{F}}$, S' must be equal to $\mathsf{g} U_1 \dots U_l$ for some λ-terms U_i such that $U_i\sigma \twoheadrightarrow_\beta V_i$. It is obvious that the head of each U_i is either $z_{\nu(i)}$ or $\mathsf{d}_{\nu(i)}$. We show that $\mathcal{F}$ is satisfied by the valuation ψ defined as follows:

$$\psi(v_i) = \begin{cases} 0 & \text{if the head of } U_i \text{ is } z_{\nu(i)} \\ 1 & \text{otherwise} \end{cases}$$

We show that each positive clause $\mathcal{C}_j \in \mathcal{F}$ is satisfied by ψ. Suppose that y_j appears in U_i. Then, the head of U_i cannot be z_k for any k, because if both y_j and z_k are in U_i, $U_i\sigma$ contains at least $(m+1)$ occurrences of f, so $U_i\sigma$ never β-reduces to V_i, which contains exactly m occurrences of f. Thus, the head of U_i is $\mathsf{d}_{\nu(i)}$ and $\psi(v_i) = 1$. Since $y_j\sigma \equiv \mathsf{fc}_j$, $U_i\sigma$ and V_i must contain c_j and this implies $\mu(i,j) = j$, i.e., $v_i \in \mathcal{C}_j$. ψ satisfies $\mathcal{C}_j$ via v_i.

We show that each negative clause $\mathcal{D}_k \in \mathcal{F}$ is satisfied by ψ. Suppose that z_k appears in U_i. Since $z_k\sigma$ contains d_k, $k = \nu(i)$ and the head of U_i is z_k. Therefore, $\psi(v_i) = 0$ and $\neg v_i \in \mathcal{D}_k$. ψ satisfies $\mathcal{D}_k$ via $\neg v_i$. □

Proposition 6.15. *Third-order interpolation modulo β ($\beta\eta$) in the linear lambda calculus is NP-complete.*

Proof. By Theorem 6.7 and Lemmas 6.13 and 6.14. □

6.5 Elimination of Constants

In this section, we show that the interpolation problem in the linear lambda calculus is NP-hard even if there are no constants, by eliminating constants in $\langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle$ in Definition 6.10.

For general unification problem, it is easy to construct a *constant-free* $\langle P^*, Q^*\rangle$ from a unification equation $\langle P, Q\rangle$, such that $\langle P, Q\rangle$ has a solution iff $\langle P^*, Q^*\rangle$ has a solution. We obtain P^* and Q^* by successive transformations performed on P and Q: first we replace the constants $\vec{\mathsf{a}}$ by fresh variables $\vec{a}$, then we replace each unknown $\mathbf{X}_i$ with $\mathbf{Y}_i\vec{a}$ and finally we abstract the free variables. If $[S_i/\mathbf{X}_i]$ is a solution for $\langle P, Q\rangle$, then $[\lambda\vec{a}.S_i^*/\mathbf{Y}_i]$ is a solution for $\langle P^*, Q^*\rangle$, where S^* is obtained by replacing each constant a by the variable a. If $[S_i^*/\mathbf{Y}_i]$ is a solution for $\langle P^*, Q^*\rangle$, then $[S_i^*\vec{\mathsf{a}}/\mathbf{X}_i]$ is a solution for $\langle P, Q\rangle$.

However, such a transformation does not work for the unification problem in the *linear* lambda calculus, because free variables can occur at most once in a linear λ-term, while constants can occur any number of times. To construct a linear interpolation equation $\langle L_{\mathcal{F}}^* \equiv \mathbf{Y}C_1^*\ldots C_m^*D_1^*\ldots D_n^*, R_{\mathcal{F}}^*\rangle$ by eliminating constants from $\langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle$ defined in Definition 6.10, we adopt the following strategy:

- Let T be among $C_1, \ldots, C_m, D_1, \ldots, D_n, R_{\mathcal{F}}$.
- Let T' be the result of replacing occurrences of each constant in T with *suitable* free variables or λ-terms constructed from free variables.
- Let $T^* \equiv \lambda\vec{x}.T'$ for a sequence $\vec{x}$ of the elements of $\mathrm{FV}(T')$.

The main issue is what variable or λ-term each occurrence of a constant should be replaced with. The formal definition and an example are given in Definition 6.16 and Example 6.17. We give the basic strategy of our transformation here. First, for each constant a, we provide an atomic type p_{a}. Second we replace the i-th occurrence of a in T with the application $x_{\mathsf{a},i}y_{\mathsf{a},i}$ of variables $x_{\mathsf{a},i}$ of type $p_{\mathsf{a}} \to \tau(\mathsf{a})$ and $y_{\mathsf{a},i}$ of type p_{a}. Finally, we close T by $\lambda\vec{x}\vec{y}$. This way, we can avoid identifying λ-terms which are surrogates for distinct constants of the same type, while preserving the well-typedness and the linearity of the interpolation equation. If $\langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle$ has a solution $[S/\mathbf{X}]$, then $\langle L^*_{\mathcal{F}}, R^*_{\mathcal{F}}\rangle$ also has a solution $[S^*/\mathbf{Y}]$ which does not essentially differ from $[S/\mathbf{X}]$. Each occurrence of a constant a in S is replaced with the application $x_{\mathsf{a},i}y_{\mathsf{a},i}$ of two bound variables of $R^*_{\mathcal{F}}$ for the appropriate i. Moreover, $[S^*/\mathbf{Y}]$ lets the arguments T^* of the unknown $\mathbf{Y}$ be applied to bound variables of $R^*_{\mathcal{F}}$ which constitute the surrogates for constants which appear in T.

A major problem is that one may construct a solution which causes a substitution of a complex λ-term for the bound variable $x_{\mathsf{a},i}$ of an argument T^* of the unknown $\mathbf{Y}$, since a λ-term which has the type $p_{\mathsf{a}} \to \tau(\mathsf{a})$ is not necessarily a bound variable $x_{\mathsf{a},j}$ in $R^*_{\mathcal{F}}$ for some j. For instance, consider the interpolation equation $\langle \mathbf{X}/\mathsf{ac}//\mathsf{b}/,\ /\mathsf{abc}/\rangle$ where $\tau(\mathsf{a}) = \tau(\mathsf{b}) = \tau(\mathsf{c}) = \mathit{str}$. Clearly it has no solution. By applying the above conversion to $\langle \mathbf{X}/\mathsf{ac}//\mathsf{b}/,\ /\mathsf{abc}/\rangle$, we obtain the following equation:

$$\langle\ \mathbf{Y}\big(\lambda x_{\mathsf{a}}^{p_{\mathsf{a}}\to \mathit{str}} y_{\mathsf{a}}^{p_{\mathsf{a}}} x_{\mathsf{c}}^{p_{\mathsf{c}}\to \mathit{str}} y_{\mathsf{c}}^{p_{\mathsf{c}}} z^{o}.x_{\mathsf{a}}y_{\mathsf{a}}(x_{\mathsf{c}}y_{\mathsf{c}}z)\big)\big(\lambda x_{\mathsf{b}}^{p_{\mathsf{b}}\to \mathit{str}} y_{\mathsf{b}}^{p_{\mathsf{b}}} z^{o}.x_{\mathsf{b}}y_{\mathsf{b}}z\big),$$
$$\lambda x_{\mathsf{a}}^{p_{\mathsf{a}}\to \mathit{str}} y_{\mathsf{a}}^{p_{\mathsf{a}}} x_{\mathsf{b}}^{p_{\mathsf{b}}\to \mathit{str}} y_{\mathsf{b}}^{p_{\mathsf{b}}} x_{\mathsf{c}}^{p_{\mathsf{c}}\to \mathit{str}} y_{\mathsf{c}}^{p_{\mathsf{c}}} z^{o}.x_{\mathsf{a}}y_{\mathsf{a}}\big(x_{\mathsf{b}}y_{\mathsf{b}}(x_{\mathsf{c}}y_{\mathsf{c}}z)\big)\ \rangle,$$

but, this has a solution

$$[\lambda w_1^{(p_{\mathsf{a}}\to \mathit{str})\to p_{\mathsf{a}}\to (p_{\mathsf{c}}\to \mathit{str})\to p_{\mathsf{c}}\to \mathit{str}} w_2^{(p_{\mathsf{b}}\to \mathit{str})\to p_{\mathsf{b}}\to \mathit{str}}.$$
$$\lambda x_{\mathsf{a}}^{p_{\mathsf{a}}\to \mathit{str}} y_{\mathsf{a}}^{p_{\mathsf{a}}} x_{\mathsf{b}}^{p_{\mathsf{b}}\to \mathit{str}} y_{\mathsf{b}}^{p_{\mathsf{b}}} x_{\mathsf{c}}^{p_{\mathsf{c}}\to \mathit{str}} y_{\mathsf{c}}^{p_{\mathsf{c}}}.w_1x_{\mathsf{a}}y_{\mathsf{a}}\big(\lambda v_{\mathsf{c}}^{p_{\mathsf{c}}} z^{o}.w_2x_{\mathsf{b}}y_{\mathsf{b}}(x_{\mathsf{c}}v_{\mathsf{c}}z)\big)y_{\mathsf{c}}\ /\ \mathbf{Y}].$$

This is the reason why we do not employ the reduction from the string rearrangement problem as in the proof of Proposition 6.9 for constant-free cases. Fortunately, we can tightly restrict λ-terms that can be substituted for bound variables in the arguments T^* of $\mathbf{Y}$, when we employ the reduction in Definition 6.10 as a basis for the new reduction. Recall that in $\langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle$ in Definition 6.10, c_j is always an argument of f, fc_j is of d_k for some k, and a λ-term whose head is d_k is of g. Hence we can let λ-terms in $\langle L^*_{\mathcal{F}}, R^*_{\mathcal{F}}\rangle$ which are surrogates for c_j have type q, surrogates for f have type $q \to r$, surrogates for d_k have type $r^m \to t$, and surrogates for g have type $t^l \to u$.

Now, we give a formal definition of the linear interpolation equation $\langle L^*_{\mathcal{F}}, R^*_{\mathcal{F}}\rangle$ for a given polarized 1N-CNF $\mathcal{F}$.

Definition 6.16. Let $\mathcal{F} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m, \mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a polarized 1N-CNF on $\mathcal{V} = \{v_1, \ldots, v_l\}$ where $\mathcal{C}_1, \ldots, \mathcal{C}_m$ are positive clauses and $\mathcal{D}_1, \ldots, \mathcal{D}_n$ are negative clauses. Two functions μ and ν are defined as in Definition 6.10. Let $\mathscr{A} = \{p_j, q, r, s_k, t, u \mid 0 \leq j \leq m,\ 1 \leq k \leq n\}$. Let $\vec{w}_{\mathcal{F}}$ be a sequence of the variables in the set

$$\{c_{i,j}^{p_{\mu(i,j)}}, \bar{c}_{i,j}^{p_{\mu(i,j)} \to q}, f_{i,j}^{q \to r}, d_{i,\nu(i)}^{s_{\nu(i)}}, \bar{d}_{i,\nu(i)}^{s_{\nu(i)} \to r^m \to t}, g^{t^l \to u} \mid 1 \leq i \leq l \text{ and } 1 \leq j \leq m\}.$$

$\langle L_{\mathcal{F}}^*, R_{\mathcal{F}}^* \rangle$ is defined by

$$L_{\mathcal{F}}^* \equiv \mathbf{Y} C_1^* \ldots C_m^* D_1^* \ldots D_n^* \quad \text{where}$$
$$\begin{cases} C_j^* \equiv \lambda f^{q \to r} \bar{c}_j^{p_j \to q} c_j^{p_j}.f(\bar{c}_j c_j) \\ D_k^* \equiv \lambda \bar{d}_k^{s_k \to r^m \to t} d_k^{s_k} f_1^{q \to r} \ldots f_m^{q \to r} x_1^q \ldots x_m^q.\bar{d}_k d_k (f_1 x_1) \ldots (f_m x_m) \\ \tau(\mathbf{Y}) = \tau(C_1^*) \to \cdots \to \tau(C_m^*) \to \tau(D_1^*) \to \cdots \to \tau(D_n^*) \to \tau(R_{\mathcal{F}}^*) \end{cases}$$
$$R_{\mathcal{F}}^* \equiv \lambda \vec{w}_{\mathcal{F}}.g^{t^l \to u} V_1^* \ldots V_l^* \quad \text{where}$$
$$V_i^* \equiv \bar{d}_{i,\nu(i)}^{s_{\nu(i)} \to r^m \to t} d_{i,\nu(i)}^{s_{\nu(i)}} \big(f_{i,1}^{q \to r} (\bar{c}_{i,1}^{p_{\mu(i,1)} \to q} c_{i,1}^{p_{\mu(i,1)}})\big) \ldots \big(f_{i,m}^{q \to r} (\bar{c}_{i,m}^{p_{\mu(i,m)} \to q} c_{i,m}^{p_{\mu(i,m)}})\big).$$

Indeed the above λ-terms are well-typed.

$$\begin{aligned} \tau(C_j^*) &= (q \to r) \to (p_j \to q) \to p_j \to r, \\ \tau(D_k^*) &= (s_k \to r^m \to t) \to s_k \to (q \to r)^m \to q^m \to t, \\ \mathrm{ord}(\mathbf{Y}) &= 4. \end{aligned}$$

Throughout this section, we use the following type assignment to variables for $1 \leq i \leq l$, $1 \leq j \leq m$ and $1 \leq k \leq n$, which is consistent with the above definition:

$$\begin{gathered} \tau(c_j) = p_j, \quad \tau(c_{i,j}) = p_{\mu(i,j)}, \quad \tau(\bar{c}_j) = p_j \to q, \quad \tau(\bar{c}_{i,j}) = p_{\mu(i,j)} \to q, \\ \tau(d_k) = \tau(d_{i,k}) = s_k, \quad \tau(\bar{d}_k) = \tau(\bar{d}_{i,k}) = s_k \to r^m \to t, \\ \tau(f) = \tau(f_j) = \tau(f_{i,j}) = q \to r, \quad \tau(g) = t^l \to u, \\ \tau(x_j) = q, \quad \tau(y_j) = \tau(C_j^*), \quad \tau(z_k) = \tau(D_k^*). \end{gathered}$$

Example 6.17. Instance $\mathcal{F} : \mathcal{C}_1 = \{v_1\},\ \mathcal{C}_2 = \{v_2\},\ \mathcal{D}_1 = \{\neg v_1, \neg v_2\}$

$$\begin{aligned}
&\text{Reduction with Constants } \langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle :\\
&\qquad L_{\mathcal{F}} \equiv \mathbf{X}(\mathsf{fc}_1)(\mathsf{fc}_2)\big(\lambda x_1 x_2.\mathsf{d}_1(\mathsf{f}x_1)(\mathsf{f}x_2)\big)\\
&\qquad R_{\mathcal{F}} \equiv \mathsf{g}\big(\mathsf{d}_1(\mathsf{fc}_1)(\mathsf{fc}_0)\big)\big(\mathsf{d}_1(\mathsf{fc}_0)(\mathsf{fc}_2)\big)\\
&\text{Reduction without Constants } \langle L^*_{\mathcal{F}}, R^*_{\mathcal{F}}\rangle :\\
&\qquad L^*_{\mathcal{F}} \equiv \mathbf{Y}\big(\lambda f \bar{c}_1 c_1.f(\bar{c}_1 c_1)\big)\big(\lambda f \bar{c}_2 c_2.f(\bar{c}_2 c_2)\big)\\
&\qquad\qquad \big(\lambda \bar{d}_1 d_1 f_1 f_2 x_1 x_2.\bar{d}_1 d_1(f_1 x_1)(f_2 x_2)\big)\\
&\qquad R^*_{\mathcal{F}} \equiv \lambda \vec{w}_{\mathcal{F}}.g\Big(\bar{d}_{1,1} d_{1,1}\big(f_{1,1}(\bar{c}_{1,1}c_{1,1})\big)\big(f_{1,2}(\bar{c}_{1,2}c_{1,2})\big)\Big)\\
&\qquad\qquad \Big(\bar{d}_{2,1} d_{2,1}\big(f_{2,1}(\bar{c}_{2,1}c_{2,1})\big)\big(f_{2,2}(\bar{c}_{2,2}c_{2,2})\big)\Big)
\end{aligned}$$

where the type of each variable is as Definition 6.16. $\mathcal{F}$ is not satisfiable and $\langle L_{\mathcal{F}}, R_{\mathcal{F}}\rangle$ and $\langle L^*_{\mathcal{F}}, R^*_{\mathcal{F}}\rangle$ have no solution. Note that

$$\begin{aligned}
[\lambda y_1 y_2 z_1 \vec{w}_{\mathcal{F}}.g\big(z_1 \bar{d}_{1,1} d_{1,1} f_{1,1} f_{1,2}(\bar{c}_{1,1}c_{1,1})(\bar{c}_{1,2}c_{1,2})\big)\\
\big(\bar{d}_{2,1} d_{2,1}(y_1 f_{2,1}\bar{c}_{2,1}c_{2,1})(y_2 f_{2,2}\bar{c}_{2,2}c_{2,2})\big)\ /\ \mathbf{Y}]
\end{aligned}$$

is not a solution for $\langle L^*_{\mathcal{F}}, R^*_{\mathcal{F}}\rangle$. $(y_1 f_{2,1}\bar{c}_{2,1}c_{2,1})$ is not a λ-term of the simply typed lambda calculus, for $\tau(y_1) = (q \to r) \to (p_1 \to q) \to p_1 \to r$ but $\tau(\bar{c}_{2,1}) = p_{\mu(2,1)} \to q = p_0 \to q$ and $\tau(c_{2,1}) = p_{\mu(2,1)} = p_0$.

Lemma 6.18. *$\langle L^*_{\mathcal{F}}, R^*_{\mathcal{F}}\rangle$ admits a solution whenever $\mathcal{F}$ is satisfiable.*

Proof. Suppose that ψ is a valuation which satisfies $\mathcal{F}$. As in the proof of Lemma 6.13, we can find $\phi : \mathcal{F} \to (\mathcal{V} \cup \neg\mathcal{V})$ which indicates a literal via which each clause is satisfied. We show that a solution is given by $[S/\mathbf{Y}]$ where

$$\begin{aligned}
&S \equiv \lambda y_1 \ldots y_m z_1 \ldots z_n \vec{w}_{\mathcal{F}}.g U_1 \ldots U_l \quad \text{where}\\
&\qquad U_i \equiv \begin{cases} z_{\nu(i)}\bar{d}_{i,\nu(i)} d_{i,\nu(i)} f_{i,1} \ldots f_{i,m}(\bar{c}_{i,1}c_{i,1}) \ldots (\bar{c}_{i,m}c_{i,m}) & \text{if } \phi(\mathcal{D}_{\nu(i)}) = \neg v_i,\\ \bar{d}_{i,\nu(i)} d_{i,\nu(i)} U_{i,1} \ldots U_{i,m} & \text{otherwise,}\end{cases}\\
&\qquad U_{i,j} \equiv \begin{cases} y_j f_{i,j}\bar{c}_{i,j}c_{i,j} & \text{if } \phi(\mathcal{C}_j) = v_i,\\ f_{i,j}(\bar{c}_{i,j}c_{i,j}) & \text{otherwise.}\end{cases}
\end{aligned}$$

Indeed S is a well-typed closed linear λ-term. The linearity of S can be checked as in the proof of Lemma 6.13. We check the well-typedness of S. Recall that $\tau(z_{\nu(i)}) = (s_{\nu(i)} \to r^m \to t) \to s_{\nu(i)} \to (q \to r)^m \to q^m \to t$, $\tau(\bar{d}_{i,\nu(i)}) = s_{\nu(i)} \to r^m \to t$, $\tau(d_{i,\nu(i)}) = s_{\nu(i)}$, $\tau(f_{i,j}) = q \to r$ and $\tau(\bar{c}_{i,j}c_{i,j}) =$

q. Hence, U_i is well-typed and has the type t if $\phi(\mathcal{D}_{\nu(i)}) = \neg v_i$. If $\phi(\mathcal{D}_{\nu(i)}) \neq \neg v_i$, it is enough to check that each $U_{i,j}$ has type r. If $\phi(\mathcal{C}_j) = v_i$ and $U_{i,j} \equiv y_j f_{i,j} \bar{c}_{i,j} c_{i,j}$, then $v_i \in \mathcal{C}_j$, $\tau(\bar{c}_{i,j}) = p_{\mu(i,j)} \to q = p_j \to q$ and $\tau(c_{i,j}) = p_{\mu(i,j)} = p_j$. Recall that $\tau(y_j) = (q \to r) \to (p_j \to q) \to p_j \to r$ and $\tau(f_{i,j}) = q \to r$. Hence, $U_{i,j}$ is well-typed and has the type r. If $\phi(\mathcal{C}_j) \neq v_i$, it is clear that $U_{i,j}$ is well-typed and has the type r. Therefore, S is well-typed.

Second we show that $L^*_{\mathcal{F}}[S/\mathbf{Y}] \twoheadrightarrow_\beta R^*_{\mathcal{F}}$. By the definition, $L^*_{\mathcal{F}}[S/\mathbf{Y}] \twoheadrightarrow_\beta \lambda \vec{w}_{\mathcal{F}}.gU_1 \dots U_l\sigma$ for $\sigma = [\vec{C}^*/\vec{y}, \vec{D}^*/\vec{y}]$. It is enough to show that $U_i\sigma \twoheadrightarrow_\beta V_i^*$. If $\phi(\mathcal{D}_{\nu(i)}) = \neg v_i$, then

$$\begin{aligned} U_i\sigma &\equiv D^*_{\nu(i)} \bar{d}_{i,\nu(i)} d_{i,\nu(i)} f_{i,1} \dots f_{i,m} (\bar{c}_{i,1} c_{i,1}) \dots (\bar{c}_{i,m} c_{i,m}) \\ &\twoheadrightarrow_\beta \bar{d}_{i,\nu(i)} d_{i,\nu(i)} \big(f_{i,1}(\bar{c}_{i,1} c_{i,1})\big) \dots \big(f_{i,m}(\bar{c}_{i,m} c_{i,m})\big) \\ &\equiv V_i^*. \end{aligned}$$

Otherwise, $U_i \equiv \bar{d}_{i,\nu(i)} d_{i,\nu(i)} U_{i,1} \dots U_{i,m}$. It is obvious that $U_{i,j}\sigma \twoheadrightarrow_\beta f_{i,j}(\bar{c}_{i,j} c_{i,j})$ for all j, since $C^*_j f_{i,j} \bar{c}_{i,j} c_{i,j} \twoheadrightarrow_\beta f_{i,j}(\bar{c}_{i,j} c_{i,j})$. □

Lemma 6.19. *Suppose that a linear λ-term M contains no free variables other than the elements of $\vec{w}_{\mathcal{F}}$ and $y_1, \dots, y_m, z_1, \dots, z_n$.*

If M has an atomic type δ and contains a subterm N of type $\gamma_1 \to \dots \to \gamma_i \to \gamma$ with $\gamma \in \mathscr{A}$, then $\gamma \leq \delta$ where $\leq$ is the partial order on $\mathscr{A}$ such that $p_j \lneq q \lneq r \lneq t \lneq u$ for $0 \leq j \leq m$ and $s_k \lneq t \lneq u$ for $1 \leq k \leq n$.

If $\tau(M) = q \to r$, then $M =_\eta f_{i,j}$ for some i and j.

Lemma 6.20. *Suppose that a subterm of a linear λ-term M has an atomic type p. Then, for every N such that $M =_\beta N$, N has a subterm of type p.*

Lemma 6.21. *$\mathcal{F}$ is satisfiable whenever $\langle L^*_{\mathcal{F}}, R^*_{\mathcal{F}} \rangle$ admits a solution.*

Proof. Suppose that $[S/\mathbf{Y}]$ is a solution. We can assume that S is β-normal and $S \equiv \lambda y_1 \dots y_m z_1 \dots z_n \vec{w}_{\mathcal{F}}.S'$. Because of the type of the variable g in $R^*_{\mathcal{F}}$, the head of S' is neither y_j nor z_k. S' must be $gU_1 \dots U_l$ for some $U_1, \dots, U_l$ of type t. Let $\sigma = [\vec{C}^*/\vec{y}, \vec{D}^*/\vec{z}]$. Since $SC^*_1 \dots C^*_m D^*_1 \dots D^*_n \twoheadrightarrow_\beta \lambda \vec{w}_{\mathcal{F}}.gU_1 \dots U_l\sigma \twoheadrightarrow_\beta \lambda \vec{w}_{\mathcal{F}}.gV^*_1 \dots V^*_l$, we have $U_i\sigma \twoheadrightarrow_\beta V_i^*$. Note that

(*) U_i contains no free variables other than the elements of

$$\begin{aligned} &\mathrm{Fv}(V_i^*) \cup \{y_j, z_k \mid 1 \leq j \leq m,\ 1 \leq k \leq n\} \\ &= \{\bar{d}_{i,\nu(i)}, d_{i,\nu(i)}, f_{i,j}, \bar{c}_{i,j}, c_{i,j}, y_j, z_k \mid 1 \leq j \leq m,\ 1 \leq k \leq n\}. \end{aligned}$$

Since U_i has type t, the head of U_i must be $\bar{d}_{i,\nu(i)}$ or z_k for some k.

First, we show that if U_i contains z_k, then z_k is the head of U_i and U_i contains neither y_j nor $z_{k'}$ for any j and $k' \neq k$. Since $z_k\sigma \equiv D^*_k$ contains a

variable of type s_k, $U_i\sigma$ and its β-normal form V_i^* contain a subterm of type s_k by Lemma 6.20. This implies that $k = \nu(i)$ and V_i^* contains no subterm of type $s_{k'}$ for any $k' \neq k$ and thus $U_i\sigma$ does so. Therefore, U_i does not contain $z_{k'}$ for any $k' \neq k$. If $U_i \equiv \bar{d}_{i,\nu(i)} M_{\nu(i)} R_1 \dots R_m$ for $\tau(M_{\nu(i)}) = s_{\nu(i)}$ and $\tau(R_h) = r$ for $1 \leq h \leq m$, then z_k cannot occur in U_i by Lemma 6.19. Therefore, if U_i contains z_k, then U_i is of the form

$$U_i \equiv z_k N_k M_k F_1 \dots F_m Q_1 \dots Q_m$$

where $\tau(N_k) = s_k \to r^m \to t$, $\tau(M_k) = s_k$, $\tau(F_h) = q \to r$, and $\tau(Q_h) = q$ for $1 \leq h \leq m$. is proved. We check that y_j cannot occur in N_k, M_k, Q_h or F_h for any h. It is obvious that y_j is not in M_k, Q_h or F_h by (*) and Lemma 6.19. Since N_k does not contain $z_{k'}$ for any k', N_k can be written as

$$N_k \equiv \lambda d_k^{s_k} w_1^r \dots w_m^r . \bar{d}_{i,\nu(i)} d_k R_1 \dots R_m$$

where $\tau(R_h) = r$ for all h (provided that $\nu(i) = k$). It is easy to see that if w_h appears in some $R_{h'}$, then $R_{h'} = w_h$. Thus the sequence $R_1 \dots R_m$ is just a permutation of $w_1 \dots w_m$. y_j does not appear in N_k. Therefore, y_j cannot occur in U_i unless the head of U_i is $\bar{d}_{i,\nu(i)}$.

Now, we show that $\mathcal{F}$ is satisfied by the valuation ψ defined as follows:

$$\psi(v_i) = \begin{cases} 0 & \text{if the head of } U_i \text{ is a variable } z_k \text{ for some } k, \\ 1 & \text{otherwise.} \end{cases}$$

By the linearity of S, each variable $y_1, \dots, y_m, z_1, \dots, z_n$ appears in U_i for some i. To show that each positive clause $\mathcal{C}_j$ is satisfied by ψ, suppose that y_j occurs in U_i. The above discussion claims that the head of U_i is $\bar{d}_{i,\nu(i)}$ and thus $\psi(v_i) = 1$. Since $y_j\sigma \equiv C_j^*$ contains a variable of type p_j, $U_i\sigma$ and its β-normal form V_i^* contain a subterm of type p_j by Lemma 6.20. Therefore, $\tau(c_{i,j}) = p_{\mu(i,j)} = p_j$ and $v_i \in \mathcal{C}_j$. Thus ψ satisfies $\mathcal{C}_j$ via v_i.

To show that each negative clause $\mathcal{D}_k$ is satisfied by ψ, suppose that z_k occurs in U_i. The above discussion claims that z_k is the head of U_i and $k = \nu(i)$. Therefore, ψ satisfies $\mathcal{D}_k$ via $\neg v_i$. □

Theorem 6.22. *Fourth-order interpolation modulo β ($\beta\eta$) in the linear lambda calculus in the absence of constants is NP-complete.*

Proof. By Theorem 6.7 and Lemmas 6.18 and 6.21. □

The order of the equation in Definition 6.16 is higher than the one in Definition 6.10 by one, though the maximum order of the types of the variables appearing in the former equation is the same as the one of the constants in

the latter. The reason of the increase of the order is due to the presence of λ-abstraction.

It is easy to see that Theorem 6.22 does not hold for the third-order case unless P = NP. For an interpolation equation $\langle \mathbf{X} L_1 \dots L_m, R \rangle$, if L_i is a constant-free closed linear λ-term whose type is at most second-order, then $L_i \equiv \lambda x_i^{p_i}.x_i$ for some atomic type $p_i \in \mathscr{A}$. Hence $\langle \mathbf{X} L_1 \dots L_m, R \rangle$ has a linear solution modulo β (*resp.* $\beta\eta$) iff (*resp.* the η-long form of) R has a subterm of type p_i for every i by Lemma 6.20.

Chapter 7

Conclusions

This thesis has investigated the mathematical properties of some extensions and restrictions of ACGs. In particular, the aspect of ACGs as a generalization of well-established grammar formalisms is investigated.

In Chapter 3, we have shown that allowing deleting operations do not enrich the expressive power of ACGs. The result entails the equivalence between linear CFTGs and non-duplicating CFTGs, and between LCFRSs and MCFGs, as corollaries. It is future work whether our procedure for elimination of vacuous λ-abstraction can be generalized so that Fisher's result [8, 9] is covered.

In Chapter 4, we have proposed a lexicalization method of semilexicalized ACGs which preserves the order of the abstract vocabulary. Our result entails that LCFRSs admit lexicalization. Moreover, our lexicalization method for second-order ACGs has a close connection with Schabes et al.'s preceding research [47, 48] that converts finitely ambiguous CFGs into lexicalized TAGs. There remains an open question, whether or not second and third-order semilexicalized ACGs can be lexicalized preserving the orders of the lexicons. If this open problem is solved in the affirmative, it would give a solution for the corresponding problems for mildly context-sensitive formalisms. To answer the question, it seems inevitable to analyze the possible forms of object terms assigned to abstract constants by the lexicon.

In Chapter 5, we have presented that several well-known tree transducer formalisms are encodable in two-dimensional ACGs. In particular, we have seen the equivalence between linear macro tree transducers and ACGs belonging to $\mathbf{G}_{\mathrm{tree}}(2, 1(\mathrm{sr}), 2)$. Therefore, the ACG formalism has the rich potential as a general framework in which other existing two-dimensional formalisms, as well as one-dimensional formalisms, may be encoded. The hierarchy of second-order two-dimensional ACGs, however, contains many subclasses whose generative capacity is not known yet, while the hierar-

chy of second-order one-dimensional ACGs has been characterized almost completely. More detailed characterization of the hierarchy of second-order two-dimensional ACGs is future work.

Those results show that the ACG formalism can be thought of as a generalization of existing grammar formalisms in various meanings. ACGs encode not only individual grammars, but also operations on them, i.e., linearization and lexicalization. Moreover, ACGs generalize two-dimensional formalisms as well as one-dimensional ones. This thesis demonstrates that investigating ACGs is to study various mathematical properties of many other well-established grammar formalisms including, but not limited to, mildly context-sensitive grammars.

On the other hand, in Chapter 6, we have proved that third-order interpolation in the linear lambda calculus is NP-complete, and moreover, fourth-order interpolation in the linear lambda calculus is NP-complete even when we exclude constants from the problem instances. Those issues are not about the ACG formalism itself, though they (in particular the former) are motivated by a topic on ACGs.

Since the ACG is a simple formalism based on simply typed linear lambda calculus, investigation of ACGs sometimes offers motivations and topics to the fields of lambda-calculus and linear logic. For instance, Salvati [43] introduces an extended type system for the linear lambda calculus, called *syntactic descriptions*, for parsing second-order ACGs, and he also proposes an algorithm for solving the linear matching problem in the linear lambda calculus that utilizes syntactic descriptions [44]. Kanazawa [25] presents a new proof for the Interpolation Theorem for the implicational fragment of intuitionistic logic and its substructural subsystems, which would be used for an algorithm that learns two-dimensional ACGs generating pairs of a string and its meaning [24]. Conversely, since the simply typed lambda calculus has been studied very well so far, we have plenty of useful tools for investigating the mathematical properties of ACGs. Indeed, the basic mathematical properties of (lexicalized) ACGs shown in Chapter 2 and 5 are directly derived from existing theorems on the intuitionistic linear logic, and moreover, other theorems presented in this thesis rely on existing theorems and techniques in the intuitionistic linear logic implicitly or explicitly. This thesis strengthens the concept that the ACG formalism might be a linkage between mathematical linguistics and logic.

Bibliography

[1] Alfred V. Aho and Jeffrey D. Ullman. Translations on a context-free grammar. *Information and Control*, 19:439–475, 1971.

[2] Brenda S. Baker. Generalized syntax-directed translation, tree transducers, and linear space. *SIAM Journal of Computing*, 7(3):376–391, August 1978.

[3] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997. release October, 1rst 2002.

[4] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, 1971. ACM.

[5] Daniel Dougherty and ToMasz Wierzbicki. A decidable variant of higher order matching. In *13th International Conference on Rewriting Techniques and Applications*, volume 2378 of *Lecture Notes in Computer Science*, pages 340–351. Springer-Verlag, 2002.

[6] Joost Engelfriet. Some open questions and recent results on tree transducers and tree languages. In R.$\tilde{\text{V}}$. Book, editor, *Formal language theory; perspectives and open problems*, pages 241–286. Academic Press, 1980.

[7] Joost Engelfriet and Heiko Vogler. Macro tree transducers. *Journal of Computer and System Science*, 31(1):71–146, 1985.

[8] Michael J. Fisher. *Grammars with Macro-Like Productions*. PhD thesis, Harvard University, 1968.

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

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

[11] Seymour Ginsburg and Sheila Greibach. Abstract families of languages. In *Studies in abstract families of languages*, *Memoirs of the American Mathematical Society*, 87:1–32, 1969.

[12] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[13] Sheila A. Greibach. A new normal-form theorem for context-free phrase structure grammars. *Journal of the ACM*, 12(1):42–52, 1965.

[14] Philippe de Groote. Linear higher-order matching is NP-complete. In *11th International Conference on Rewriting Techniques and Applications*, volume 1833 of *Lecture Notes in Computer Science*, pages 127–140. Springer-Verlag, 2000.

[15] Philippe de Groote. 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.

[16] Philippe de Groote. 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, 2002.

[17] Philippe de Groote, 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, July 2004.

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

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

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

[21] John E. Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135–159, 1979.

[22] Aravind K. Joshi, K. Vijay-Shanker, and David J. Weir. The convergence of mildly context-sensitive grammar formalisms. In P. Sells, S. M. Shieber, and T. Wasow, editors, *Foundational Issues in Natural Language Processing*, pages 31–81. MIT Press, Cambridge, MA, 1991.

[23] Tsutomu Kamimura and Giora Slutzki. Parallel and two-way automata on directed ordered acyclic graphs. *Information and Control*, 49(1):10–51, 1981.

[24] Makoto Kanazawa. Computing word meanings by interpolation. In *Proceedings of the Fourteenth Amsterdam Colloquium*, pages 157–162. ILLC/Department of Philosophy, University of Amsterdam, 2003.

[25] Makoto Kanazawa. Computing interpolants in implicational logics. *Annals of Pure and Applied Logic*, 142:125–201, 2006.

[26] Makoto Kanazawa. Abstract families of abstract categorial languages. In *Proceedings of 13th Workshop on Logic, Language, Information and Computation*, Electronic Notes in Theoretical Computer Science, 2006, to appear.

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

[28] Max I. Kanovich. The complexity of Horn fragments of linear logic. *Annals of Pure and Applied Logic*, 69:195–241, 1994.

[29] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

[30] Jordi Levy. Linear second-order unification. In *7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 332–346. Springer-Verlag, 1996.

[31] Richard J. Lipton. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University, January 1976.

[32] Ralph Loader. Higher order β matching is undecidable. *Logic Journal of the IGPL*, 11(1):51–68, 2003.

[33] Jens Michaelis. Derivational minimalism is mildly context-sensitive. In *Proceedings of the Third International Conference on Logical Aspects of Computational Linguistics*, pages 179–198, London, UK, 1998. Springer-Verlag.

[34] Jens Michaelis. Transforming linear context-free rewriting systems into minimalist grammars. In *Proceedings of the Fourth International Conference on Logical Aspects of Computational Linguistics*, pages 228–244, London, UK, 2001. Springer-Verlag.

[35] Michael Moortgat. In situ binding: A modal analysis. In *Proceedings of Tenth Amsterdam Colloquium*, pages 539–549, 1995.

[36] Michael Moortgat. Categorial type logics. In Johan van Benthem and A ter Meulen, editors, *Handbook of Logic and Language*, pages 93–177. Elsevier, 1997.

[37] Glyn V. Morrill. *Type Logical Grammar: Categorial Logic of Signs*. Kluwer Academic Publishers, Dordrecht, 1994.

[38] Reinhard Muskens. Language, lambdas, and logic. In Geert-Jan Kruijff and Richard Oehrle, editors, *Resource Sensitivity in Binding and Anaphora*, Studies in Linguistics and Philosophy, pages 23–54. Kluwer, 2003.

[39] M. Nivat. Transductions des langages de chomsky. In *Annales*, volume 18, pages 339–456. Institut Fourier, 1968.

[40] Richard T. Oehrle. Term-labeled categorial type systems. *Linguistics and Philosophy*, 17(6):633–678, 1994.

[41] Richard T. Oehrle. Some 3-dimensional systems of labelled deduction. *Bulletin of the Interest Group in Pure and Applied Logics*, 3(2-3):429–448, 1995.

[42] Sylvain Pogodalla. Using and extending ACG technology: Endowing categorial grammars with an underspecified semantic representation. In *Proceedings of Categorial Grammars 2004, Montpellier*, pages 197–209, June 2004.

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

[44] Sylvain Salvati. Syntactic descriptions: a type system for solving matching equations in the linear λ-calculus. In *17th International Conference on Rewriting Techniques and Applications*, volume 4098 of *Lecture Notes in Computer Science*, pages 151–165. Springer-Verlag, 2006.

[45] Sylvain Salvati. Encoding second order string ACG with deterministic tree walking transducers. In *Proceedings of the 11th conference on Formal Grammar*, CSLI Publications Online Proceedings, pages 119–131, 2006.

[46] Sylvain Salvati and Philippe de Groote. On the complexity of higher-order matching in the linear λ-calculus. In *14th International Conference on Rewriting Techniques and Applications*, volume 2706 of *Lecture Notes in Computer Science*, pages 234–245. Springer-Verlag, 2003.

[47] Yves Schabes. *Mathematical and Computational Aspects Of Lexicalized Grammars*. PhD thesis, Department of Computer and Information Science, School of Engineering and Applied Science, University of Pennsylvania, 1990.

[48] Yves Schabes, Anne Abeillé, and Joshi Aravind K.˙ Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 578–583. Association for Computational Linguistics, 1988.

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

[50] Stuart M. Shieber. Restricting the weak-generative capacity of synchronous tree-adjoining grammars. In *Proceedings of the Second TAG Workshop*, University of Pennsylvania, Philadelphia, Pennsylvania, June 24–26 1992.

[51] Stuart M. Shieber. Synchronous grammars as tree transducers. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+ 7)*, Vancouver, Canada, May 20-22 2004.

[52] Stuart M. Shieber. Unifying synchronous tree-adjoining grammars and tree transducers via bimorphisms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, Trento, Italy, 3–7 April 2006.

[53] Stuart M. Shieber and Yves Schabes. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, volume 3, pages 253–258, 1990.

[54] Colin Stirling. A game-theoretic approach to deciding higher-order matching. In *Part II of Proceedings of 33rd International Colloquium on Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 348–359. Springer-Verlag, 2006.

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

[56] David J. Weir. *Characterizing mildly context-sensitive grammar formalisms*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 1988. Supervisor-Aravind K. Joshi.

[57] David J. Weir. Linear context-free rewriting systems and deterministic tree-walking transducers. In *Meeting of the Association for Computational Linguistics*, pages 136–143, 1992.

[58] Camille Yamada. A study of abstract categorial grammars: First-order restriction and strong generative capacity. Master's thesis, University of Tokyo, 2005.

[59] Ryo Yoshinaka. Higher-order matching in the linear lambda calculus in the absence of constants is NP-complete. In *Proceedings of the 16th International Conference on Rewriting Techniques and Appications*, volume 3467 of *Lecture Notes in Computer Science*, pages 235–249. Springer-Verlag, 2005.

[60] Ryo Yoshinaka. Linearization of affine abstract categorial grammars. In *Proceedings of the 11th conference on Formal Grammar*, CSLI Publications Online Proceedings, pages 161–175, 2006.

[61] Ryo Yoshinaka and Makoto Kanazawa. The complexity and generative capacity of lexicalized abstract categorial grammars. In *Proceedings of the 5th International Conference on Logical Aspects of Computational Linguistics*, volume 3492 of *Lecture Notes in Artificial Intelligence*, pages 330–346. Springer-Verlag, 2005.