Journal of Logic, Language and Information **13:** 421–438, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

On the Expressive Power of Abstract Categorial Grammars: Representing Context-Free Formalisms

PHILIPPE DE GROOTE and SYLVAIN POGODALLA
INRIA Lorraine, 615 rue du Jardin Botanique, B. P. 101, 54602 Villers-lès-Nancy Cedex, France

(Received 7 June 2004)

Abstract. We show how to encode context-free string grammars, linear context-free tree grammars, and linear context-free rewriting systems as Abstract Categorial Grammars. These three encodings share the same constructs, the only difference being the interpretation of the composition of the production rules. It is interpreted as a first-order operation in the case of context-free string grammars, as a second-order operation in the case of linear context-free tree grammars, and as a third-order operation in the case of linear context-free rewriting systems. This suggest the possibility of defining an Abstract Categorial Hierarchy.

Key words: Abstract categorial grammars, context-free grammars, formal language theory, lambda-calculus

1. Introduction

Abstract Categorial Grammars (ACGs) (de Groote, 2001) are a new categorial formalism based on Girard linear logic (Girard, 1987). This formalism, which sticks to the spirit of current type-logical grammars (Carpenter, 1996; Moortgat, 1997; Morrill, 1994; Oehrle, 1994), offers the following features:

- Every ACG generates two languages, an abstract language and an object language. The abstract language may be seen as a set of abstract grammatical structures, and the object language as the set of concrete forms generated from these abstract structures. Consequently, one has a direct control on the parse structures of the grammar.
- The languages generated by the ACGs are sets of linear λ-terms, which generalizes both string-languages and tree-languages.
- ACGs are based on a small set of mathematical primitives that combine via simple composition rules. Consequently, ACGs offer a rather flexible framework.

Abstract Categorial Grammars are not intended to be yet another grammatical formalism that would compete with other well-established formalisms. They should rather be seen as the kernel of a grammatical framework – in the spirit of (Ranta,

2004) – in which other existing grammatical models may be encoded. In this paper, we illustrate this fact by exploring the expressive power of ACGs. We show how to encode three context-free formalisms (namely, context-free string grammars, linear context-free tree grammars, and linear context-free rewriting systems) as ACGs.

The paper is organized as follows. In the next section, we introduce the notion of Abstract Categorial Grammar. Section 3 gives a natural encoding of strings as linear λ-terms. In Section 4, we remind the reader of the definitions of a context-free string grammar, a linear context-free tree grammar, and a linear context-free rewriting system. In Sections 5, we explain how to encode context-free derivations. Then, Sections 6–8 give the encodings of context-free string grammars, linear context-free tree grammars, and linear context-free rewriting systems, respectively. Finally, we conclude in Section 9.

2. Abstract Categorial Grammars

This section gives the definition of an Abstract Categorial Grammar, which is based on the notions of *linear implicative types, higher-order linear signature*, and *linear λ-terms* built upon a higher-order linear signature.

Let A be a set of atomic types. The set $\mathcal{T}(A)$ of *linear implicative types* built upon A is inductively defined as follows:

1. if $a \in A$, then $a \in \mathcal{T}(A)$;
2. if $\alpha, \beta \in \mathcal{T}(A)$, then $(\alpha \multimap \beta) \in \mathcal{T}(A)$.

We use the usual convention of right association of the parentheses, i.e., we write $\alpha \multimap \beta \multimap \gamma \multimap \delta$ for $(\alpha \multimap (\beta \multimap (\gamma \multimap \delta)))$. We also write $\alpha^n \multimap \beta$ for

$$\underbrace{\alpha \multimap \cdots \multimap \alpha}_{n\times} \multimap \beta.$$

A *higher-order linear signature* consists of a triple $\Sigma = \langle A, C, \tau \rangle$, where:

1. A is a finite set of atomic types;
2. C is a finite set of constants;
3. $\tau : C \to \mathcal{T}(A)$ is a function that assigns to each constant in C a linear implicative type in $\mathcal{T}(A)$.

Let X be an infinite countable set of λ-variables. The set $\Lambda(\Sigma)$ of *linear λ-terms* built upon a higher-order linear signature $\Sigma = \langle A, C, \tau \rangle$ is inductively defined as follows:

1. if $c \in C$, then $c \in \Lambda(\Sigma)$;
2. if $x \in X$, then $x \in \Lambda(\Sigma)$;

3. if $x \in X$, $t \in \Lambda(\Sigma)$, and x occurs free in t exactly once, then $(\lambda x.t) \in \Lambda(\Sigma)$;
4. if $t, u \in \Lambda(\Sigma)$, and the sets of free variables of t and u are disjoint, then $(tu) \in \Lambda(\Sigma)$.

$\Lambda(\Sigma)$ is provided with the usual notion of capture avoiding substitution, and the relations of α-conversion, β-reduction, β-conversion, and $\beta\eta$-conversion (Barendregt, 1984), this latter relation being used as the notion of equality between λ-terms. We use the usual conventions when writing λ-terms: $t\ u_1 u_2 \ldots u_n$ will stand for $(\ldots((t\ u_1)u_2)\ldots u_n)$, and $\lambda x_1 \ldots x_n.t$ for $\lambda x_1 \ldots .\lambda x_n.t$. Moreover, when $\mathbf{x}$ denotes a sequence of λ-variables $x_1, \ldots, x_n$, We write $\lambda\mathbf{x}.t$ for $\lambda x_1 \ldots x_n.t$.

Given a higher-order linear signature $\Sigma = \langle A, C, \tau \rangle$, each linear λ-term in $\Lambda(\Sigma)$ may be assigned a linear implicative type in $\mathscr{T}(A)$. This type assignment obeys an inference system whose judgements are sequents of the following form:

$$\Gamma \vdash_\Sigma t : \alpha$$

where:

1. Γ is a finite set of λ-variable typing declarations of the form '$x : \beta$' (with $x \in X$ and $\beta \in \mathscr{T}(A)$), such that any λ-variable is declared at most once;
2. $t \in \Lambda(\Sigma)$;
3. $\alpha \in \mathscr{T}(A)$.

The axioms and inference rules are the following:

$$\vdash_\Sigma c : \tau(c) \qquad \text{(cons)}$$

$$x : \alpha\lambda x : \alpha \qquad \text{(var)}$$

$$\frac{\Gamma, x : \alpha\ t : \beta}{\Gamma \vdash_\Sigma (\lambda x.\ t) : (\alpha \multimap \beta)} \qquad \text{(abs)}$$

$$\frac{\Gamma \vdash_\Sigma t : (\alpha \multimap \beta)\ \Delta \vdash_\Sigma u : \alpha}{\Gamma, \Delta \vdash_\Sigma (t\ u) : \beta} \qquad \text{(app)}$$

Given two higher-order linear signatures Σ_1 and Σ_2, we define a *lexicon* $\mathscr{L} : \Sigma_1 \to \Sigma_2$ to be a realization of Σ_1 into Σ_2 i.e., an interpretation of the atomic types of Σ_1 as types built upon Σ_2 together with an interpretation of the constants of Σ_1 as linear λ-terms built upon Σ_2. These two interpretations must be such that their homomorphic extensions commute with the typing relations. This is spelled out in the next definition.

DEFINITION 1. Let $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ be two higher-order linear signatures. A lexicon $\mathscr{L}$ from Σ_1 to Σ_2 is defined to be a pair $\mathscr{L} = \langle F, G \rangle$ such that:

1. $F : A_1 \to \mathcal{T}(A_2)$ is a function that interprets the atomic types of Σ_1 as linear implicative types built upon A_2;
2. $G : C_1 \to \Lambda(\Sigma_2)$ is a function that interprets the constants of Σ_1 as linear λ-terms built upon Σ_2;
3. the interpretation functions are compatible with the typing relation, i.e., for any $c \in C_1$, the following typing judgement is derivable:

$$\vdash_{\Sigma_2} G(c) : \hat{F}(\tau_1(c)),$$

where $\hat{F}$ is the unique homomorphic extension of F.

In the sequel, given such a lexicon $\mathcal{L} = \langle F, G\rangle$, $\mathcal{L}(a)$ will stand for either $\hat{F}(a)$ or $\hat{G}(a)$, according to the context.

We are now in a position of defining the notion of Abstract Categorial Grammar.

DEFINITION 2. An Abstract Categorial Grammar is a quadruple $\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s\rangle$ where:

1. Σ_1 and Σ_2 are two higher-order linear signatures; they are called the abstract vocabulary and the object vocabulary, respectively;
2. $\mathcal{L} : \Sigma_1 \to \Sigma_2$ is a lexicon from the abstract vocabulary to the object vocabulary;
3. s is an atomic type of the abstract vocabulary; it is called the distinguished type of the grammar.

Every ACG $\mathcal{G}$ generates two languages: an *abstract language, $\mathcal{A}(\mathcal{G})$*, and an object language $\mathcal{O}(\mathcal{G})$.

The abstract language, which may be seen as a set of abstract parse structures, is the set of closed linear λ-terms built upon the abstract vocabulary and whose type is the distinguished type of the grammar.

DEFINITION 3. Let $\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s\rangle$ be an Abstract Categorial Grammar. The abstract language $\mathcal{A}(\mathcal{G})$, generated by $\mathcal{G}$ is defined as follows:

$$\mathcal{A}(\mathcal{G}) = \{t \in \Lambda(\Sigma_1) \mid \vdash_{\Sigma_1} t : s \text{ is derivable}\}$$

On the other hand, the *object language,* which may be seen as the set of concrete forms generated by the grammar, is defined to be the image of the abstract language by the term homomorphism induced by the lexicon.

DEFINITION 4. Let $\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s\rangle$ be an Abstract Categorial Grammar. The object language $\mathcal{O}(\mathcal{G})$, generated by $\mathcal{G}$ is defined as follows:

$$\mathcal{O}(\mathcal{G}) = \{t \in \Lambda(\Sigma_2) \mid \exists u \in \mathcal{A}(\mathcal{G}).\, t = \mathcal{L}(u)\}$$

3. Strings as Linear λ-Terms

We are concerned, in this paper, with the representation of grammatical formalisms that generate strings. We must, therefore, specify a higher-order linear signature that allows strings to be defined and manipulated. This signature will serve as the object vocabulary of the several ACGs we will define.

There is, in fact, a canonical way of representing strings as linear λ-terms. It consists of encoding a string of symbols as a composition of functions. Consider, for instance, a string such as *'abbac'*. It may be represented by the linear λ-term:

$$\lambda x.a(b(b(a(cx)))),$$

where the atomic strings *'a', 'b',* and *'c'* are declared to be constants of functional type.

More formally, the higher-order linear signature corresponding to an alphabet obeys the following definition.

DEFINITION 5. let $T = \{a_1, \ldots, a_n\}$ be an alphabet. The higher-order linear signature, $\Sigma_T = \langle A, C, \tau \rangle$, is defined as follows:

1. $A = \{\sigma\}$;
2. $C = \{a_1, \ldots, a_n\}$;
3. $\tau(a_i) = (\sigma \multimap \sigma)$, for all $1 \le i \le n$.

Given such a signature, the empty word (ϵ) is represented by the identity function ($\lambda x.x$), and concatenation is defined to be functional composition ($\lambda f.\lambda g.\lambda x.f(g\ x)$), which is indeed an associative operator that admits the identity function as a unit.

We define *string* to be the type ($\sigma \multimap \sigma$), and λ-terms of type *string,* such as $\lambda x.a(b(b(a(c\ x))))$, will be written */abbac/*. Finally, the infix operator $+$ will denote the composition (i.e., the concatenation) of such λ-terms.

4. Three Context-Free Formalisms

In this section, we remind the reader of the definitions of the grammatical formalisms we intend to encode as ACGs.

4.1. Context-Free String Grammars

A context-free string grammar is a quadruple $G = \langle N, T, P, s \rangle$ where:

1. *N* is a finite set of symbols called the alphabet of non-terminal symbols;
2. *T* is a finite set of symbols, disjoint from *N,* called the alphabet of terminal symbols;

3. P is a finite set of production rules of the form $a \rightarrow \alpha$, where $a \in N$, and $\alpha \in (N \cup T)^*$;
4. $s \in N$ is called the start symbol of the grammar.

Given two words $\alpha, \beta \in (N \cup T)^*$, one says that β is directly derivable from α if and only if there exist $\beta_1, \beta_2, \beta_3 \in (N \cup T)^*$ and $a \in N$ such that:

1. $a \rightarrow \beta_2$ is a production rule of *P;*
2. $\alpha = \beta_1 a \beta_3$;
3. $\beta = \beta_1 \beta_2 \beta_3$.

This relation of direct derivability is written $\alpha \Rightarrow \beta$ and, as usual, $\Rightarrow^*$ denotes the reflexive, transitive closure of $\Rightarrow$. Finally, the language generated by G is defined to be the set of terminal words $\alpha \in T^*$ such that $s \Rightarrow^* \alpha$.

4.2. Linear Context-Free Tree Grammars

A ranked alphabet is defined to be a pair $\Sigma = \langle F_\Sigma, r_\Sigma \rangle$ such that F_Σ is a finite set of symbols, and $r_\Sigma : F_\Sigma \rightarrow \mathbb{N}$ is a function that assigns to each symbol a natural number called its rank. By a slight abuse of notation, we will write $a \in \Sigma$ for $a \in F_\Sigma$.

Given such a ranked alphabet Σ, and a possibly inifinite countable set of variables X, the set of trees $T_\Sigma(X)$ is inductively defined as follows:

1. $X \subset T_\Sigma(X)$;
2. if $f \in \Sigma$ and $r_\Sigma(f) = 0$ then $f \in T_\Sigma(X)$;
3. if $f \in \Sigma$, $r_\Sigma(f) = n$, and $t_1, \ldots, t_n \in T_\Sigma(X)$ then $f(t_1, \ldots, t_n) \in T_\Sigma(X)$.

In case X is the empty set, the set of trees $T_\Sigma\ (\phi)$ is simply written T_Σ. Let $X_n = \{x_1, \ldots, x_n\}$ be a finite set of variables. A tree $t \in T_\Sigma(X_n)$ that contains exactly one occurrence of each variable x_i $(1 \leq i \leq n)$ is called an n-context. Let t be such an n-context, and let $u_1, \ldots, u_n \in T_\Sigma$. We write $t[u_1, \ldots, u_n]$ to denote the tree obtained from t by replacing $x_1, \ldots, x_n$ by $u_1, \ldots, u_n$, respectively. The set of n-contexts built upon a given ranked alphabet Σ will be written $C_\Sigma(n)$. Strictly speaking, the notion of n-context should not depend on the choice of the set X_n. Nevertheless, in the sequel, we will use the following convention: if t is an n-context then t and $t[x_1, \ldots, x_n]$ denote the same tree.

Let X be a set of variables, let Σ be a ranked alphabet, and let Σ_0 be the set of symbols $a \in \Sigma$ such that $r_\Sigma(a) = 0$. To each tree $t \in T_\Sigma(X)$, one associates its yield $\overline{t}$, which is a string over Σ_0 inductively defined as follows:

1. $\overline{x} = x$, for $x \in X$;
2. $\overline{a} = a$, for $a \in \Sigma_0$;
3. $\overline{f(t_1, \ldots, t_n)} = \overline{t_1} \ldots \overline{t_n}$.

A linear context-free tree grammar is a quadruple $G = \langle N, T, P, s\rangle$ where:

1. N is a ranked alphabet of non-terminal symbols;
2. T is a ranked alphabet of terminal symbols, disjoint from N;
3. P is a finite set of production rules of the form $a(x_1, \ldots, x_n) \rightarrow t[x_1, \ldots, x_n]$, where $a \in N, r_N(a) = n$, the variable $x_1, \ldots, x_n$ are all distinct, and $t \in C_{N\cup T}(n)$.
4. the start symbol $s \in N$ is such that $r_N(s) = 0$.

Let $u, v \in T_{N\cup T}$. v is directly derivable from u ($u \Rightarrow v$) if and only if there exist $c \in C_{N\cup T}(1)$, $a \in N$ with $r_N(a) = n, t \in C_{N\cup T}(n)$ and $u_1, \ldots, u_n \in T_{N\cup T}$ such that:

1. $a(x_1, \ldots, x_n) \rightarrow t[x_1, \ldots, x_n]$ is a production rule of P;
2. $u = c[a(u_1, \ldots, u_n)]$;
3. $v = c[t[u_1, \ldots, u_n]]$.

The tree language generated by G is then defined to be the set of terminal trees $t \in T_T$ such that $s \Rightarrow^* t$, where $\Rightarrow^*$ stands for the reflexive, transitive closure of $\Rightarrow$.

Note that the tree language generated by a linear context-free tree grammar is not sensitive to the derivation mode. This is due to the linearity condition which derives from the fact that the right-hand side of a production rule is restricted to be a context rather than an arbitrary tree. Consequently, the usual distinction between *outside-in* and *inside-out* tree languages does not apply in the present case.

In this paper, we are interested in string languages rather than in tree languages. Consequently, we will focus on the yield language generated by a linear context-free tree grammar, i.e., the set of strings α such that $\alpha = \bar{t}$ for some tree $t \in T_T$ such that $s \Rightarrow^* t$. In the general case, the class of yield languages generated by the context-free tree grammars corresponds to the class of indexed languages. In our case, because of the linearity constraint, the class of yield languages we consider is much more restrictive. To the best of our knowledge, whether this class corresponds to a class of languages definable by some other well-established formalism is an open question. Nevertheless, it is worth noting that it contains Joshi's Tree Adjoining Languages (Joshi and Schabes, 1997) as a proper subclass (Mönnich, 1997).

4.3. Linear Context-Free Rewriting Systems

Linear context-free rewriting systems (Vijay-Shanker et al., 1987; Weir, 1988) may be defined as a proper subclass of multiple context-free grammars (Seki et al., 1991), which are themeselves a particular case of generalized context-free grammars (Pollard, 1984). We do not follow this general approach here, but give a direct tailor-made definition, which is indeed equivalent to Weir's.

Let T be an alphabet, and consider a function $f : (T^*)^m \to (T^*)^n$ that acts on tuples of strings. Such a function is called a linear transform if and only if there exist

$$\alpha_{10}, \alpha_{11}, \ldots, \alpha_{1p_1}, \ldots, \alpha_{n0}, \alpha_{n1}, \ldots, \alpha_{np_n} \in T^*$$

such that:

$$f\langle x_1, \ldots, x_m\rangle = \langle \alpha_{10}x_{11}\alpha_{11}\ldots x_{1p_1}\alpha_{1p_1}, \ldots, \alpha_{n0}x_{n1}\alpha_{n1}\ldots x_{np_n}\alpha_{np_n}\rangle$$

where $\bigcup_{i=1}^{m}\{x_i\} = \bigcup_{i=1}^{n}{}_{j=1}^{p_i}\{x_{ij}\}$, and $x_{ij} \neq x_{kl}$, whenever $i \neq k$ or $j \neq l$.

In the sequel, we work modulo the associativity of the cartesian product, i.e., we identify $(T^*)^n \times (T^*)^m$ with $(T^*)^{n+m}$ and, consequently, $\langle\langle \alpha_1, \ldots, \alpha_n\rangle, \langle \beta_1, \ldots, \beta_m\rangle\rangle$ with $\langle \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m\rangle$.

A linear context-free rewriting system is defined to be a quadruple $G = \langle N, T, P, s\rangle$ where:

1. N is a ranked alphabet of non-terminal symbols;
2. T is an alphabet of terminal symbols, disjoint from N;
3. P is a finite set of production rules of the form $\langle f, a \to \alpha\rangle$ where:
   (a) $a \in N$,
   (b) $\alpha = a_1 \ldots a_n \in N^*$,
   (c) f is a linear transform from $(T^*)^{\Sigma_{i=1}^{n} r_N(a_i)}$ into $(T^*)^{r_N(a)}$,
4. the start symbol $s \in N$ is such that $r_N(s) = 1$.

In Clause 3, the non-terminal word α is possibly empty, in which case the linear transform f degenerates into a constant tuple $f\langle\rangle$.

To each non-terminal symbol $a \in N$, one associates a set $L(a) \subset (T^*)^{r_N(a)}$, inductively defined as follows:

1. for each production rule $\langle f, a \to \epsilon\rangle$, where ϵ stands for the empty word, one has $f\langle\rangle \in L(a)$;
2. If $t_1 \in L(a_1), \ldots, t_n \in L(a_n)$, and $\langle f, a \to a_1 \ldots a_n\rangle$ is a production rule of P, then $f\langle t_1, \ldots, t_n\rangle \in L(a)$.

The language generated by G is then defined to be the set *L(s)*. Observe that this set is indeed a set of strings because $r_N(s) = 1$.

5. Specifying Context-Free Derivations

In order to encode a formalism as an ACG, we have to give an abstract vocabulary, an object vocabulary, and a lexicon. The three formalisms of Section 4 generate string languages. Consequently, their object vocabulary will obey the construct of Definition 5. They will also share the same kind of abstract vocabulary, whose construction is explained in the present section.

Let $a \rightarrow \alpha$ be a production rule of a context-free string grammar. We define the skeleton of this rule to be the pair $\langle a, \lceil \alpha \rceil \rangle$, where $\lceil \alpha \rceil$ is a word of non-terminal symbols inductively defined as follows:

1. $\lceil b \rceil = \epsilon$, if b is a terminal symbol;
2. $\lceil b \rceil = b$, if b is a non-terminal symbol;
3. $\lceil b\beta \rceil = \lceil \beta \rceil$, if b is a terminal symbol;
4. $\lceil b\beta \rceil = b\lceil \beta \rceil$, if b is a non-terminal symbol.

Similarly, let $a(x_1, \ldots, x_n) \rightarrow t$ be a production rule of a context-free tree grammar. Its skeleton is defined to be the pair $\langle a, \lceil t \rceil \rangle$, where $\lceil t \rceil$ is inductively defined as follows:

1. $\lceil x_i \rceil = \epsilon$, for x_i a variable;
2. $\lceil f \rceil = \epsilon$, if *f* is a terminal symbol of rank 0;
3. $\lceil f \rceil = f$, if *f* is a non-terminal symbol of rank 0;
4. $\lceil f(t_1, \ldots, t_n) \rceil = \lceil t_1 \rceil \ldots \lceil t_n \rceil$, if *f* is a terminal symbol;
5. $\lceil f(t_1, \ldots, t_n) \rceil = f\lceil t_1 \rceil \ldots \lceil t_n \rceil$, if *f* is a non-terminal symbol.

Finally, let $\langle f, a \rightarrow \alpha \rangle$ be a production rule of a linear context-free rewriting system. Its skeleton is defined to be the pair $\langle a, \alpha \rangle$.

To summarize, in the three cases, the skeleton of a production rule is a pair $\langle a, \alpha \rangle$, where a is the non-terminal symbol occurring in the left-hand side of the rule, and α is a word consisting of the non-terminal symbols occurring in its right-hand side.

This notion of skeleton of a production rule allows us to define the higher-order linear signature associated to a given context-free string grammar, linear context-free tree grammar, or linear context-free rewriting system.

DEFINITION 6. Let $G = \langle N, T, P, s \rangle$ be a context-free string grammar, a linear context-free tree grammar, or a linear context-free rewriting system. The higher-order linear signature $\Sigma_G = \langle A, C, \tau \rangle$, associated to G, is defined as follows:

1. $A = N$;
2. to each $p \in P$, one associates a constant c_p, and $C = \bigcup_{p \in P} \{c_p\}$;
3. $\tau(c_p) = a_1 \multimap \ldots a_n \multimap a$, where $\langle a, a_1 \ldots a_n \rangle$ is the skeleton of rule p.

It is not difficult to see that the closed λ-terms of atomic type built upon the above signature are regular trees that correspond to context-free parse trees.

6. Composition as First-Order Susbtitution

In order to define ACGs representing the formalisms of Section 4, it remains to specify appropriate lexicons. This section explains the construction of such lexicons in the case of context-free string grammars.

Let $G = \langle N, T, P, s\rangle$ be a context-free string grammar, and let $p \in P$ be the following production rule:

$$a \to \alpha_0 a_1 \alpha_1 \ldots a_n \alpha_n$$

where $a_1, a_1 \ldots, a_n \in N$ and $\alpha_0, \alpha_1, \ldots, \alpha_n \in T^*$. The linear λ-term $[\![p]\!]$ is defined to be:

$$\lambda y_1 \ldots y_n . /\alpha_0/ + y_1 + /\alpha_1/ + \cdots + y_n + /\alpha_n/$$

We now define the ACG corresponding to a given context-free string grammar.

DEFINITION 7. Let $G = \langle N, T, P, s\rangle$ be a context-free string grammar. The Abstract Categorial Grammar $\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s\rangle$ is defined as follows:

1. the abstract vocabulary Σ_G is constructed according to Definition 6;
2. the object vocabulary Σ_T is constructed according to Definition 5;
3. the lexicon $\mathcal{L}_G : \Sigma_G \to \Sigma_T$ is such that:

   (a) $\mathcal{L}_G(a) = \mathit{string}$, for all $a \in N$
   (b) $\mathcal{L}_G(c_p) = [\![p]\!]$ for all $p \in P$;

4. the distinguished type s is identical to the start symbol of G.

It remains to prove that the ACG constructed according to the above definition is indeed a correct representation of the corresponding context-free string grammar. This is established by the next two propositions.

PROPOSITION 1. *Let $G = \langle N, T, P, s\rangle$ be a context-free string grammar, and let $\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s\rangle$ be the Abstract Categorial Grammar constructed from G according to Definition 7.*

For all $a \in N$ and all $\alpha \in T^*$, if $a \Rightarrow^* \alpha$ then there exists a closed λ-term $t \in \Lambda(\Sigma_G)$ such that $\vdash_{\Sigma_G} t : a$ and $\mathcal{L}_G(t) = /\alpha/$.

Proof. We proceed by induction on the length of the derivation $a \Rightarrow^* \alpha$.

If $a \Rightarrow^* \alpha$ because of a production rule $a \to \alpha$, there must exist an abstract constant c corresponding to this production rule, which is of type a and such that $\mathcal{L}_G(c) = /\alpha/$.

Now, suppose that the first rule of the derivation is

$$a \rightarrow \alpha_0 a_1 \alpha_1 \ldots a_n \alpha_n \tag{1}$$

Consequently, there exists $\beta_1, \ldots, \beta_n \in T^*$ such that $a_i \Rightarrow^* \beta_i$ and $\alpha = \alpha_0 \beta_1 \alpha_1 \ldots \beta_n \alpha_n$. Then, by induction hypothesis, there must exist closed λ-terms $t_1, \ldots, t_n$ of type $a_1, \ldots, a_n$, respectively, such that $\mathscr{L}_G(t_i) = /\beta_i/$. On the other hand, there exists an abstract constant c corresponding to (1), whose type is $a_1 \multimap \ldots a_n \multimap a$ and such that

$$\mathscr{L}_G(c) = \lambda y_1 \ldots y_n./\alpha_0/ + y_1 + /\alpha_1/ + \cdots + y_n + /\alpha_n/.$$

Consequently, we have that

$$\mathscr{L}_G(ct_1 \ldots t_n) = /\alpha_0/ + /\beta_1/ + /\alpha_1/ + \cdots + /\beta_n/ + /\alpha_n/ = /\alpha/.$$

□

PROPOSITION 2. *Let $\mathscr{G}_G = \langle \Sigma_G, \Sigma_T, \mathscr{L}_G, s \rangle$ be the Abstract Categorial Grammar constructed from a given context-free string grammar $G = \langle N, T, P, s \rangle$, according to Definition 7.*

For all $a \in N$, and all closed λ-term $t \in \Lambda(\Sigma_G)$ such that $\vdash_{\Sigma_G} t : a$, there exists $\alpha \in T^*$ such that $\mathscr{L}_G(t) = /\alpha/$, and $a \Rightarrow^* \alpha$.

Proof. We proceed by induction on the structure of *t*. Note that, *t* being a closed term of atomic type, it is either a constant or an application.

If *t* is a constant then $t = c_p$ for some $p \in P$ whose skeleton is $\langle a, \epsilon \rangle$. Then, by definition of $\mathscr{G}_G$, p must be of the form $a \rightarrow \alpha$ with $\mathscr{L}_G(c_p) = /\alpha/$.

If *t* is an application then $t = c_p t_1, \ldots, t_n$ for some $p \in P$ whose skeleton is $\langle a, a_1, \ldots, a_n \rangle$. In this case, each λ-term t_i must be a closed λ-term of type a_i, and p must be of the form,

$$a \rightarrow \alpha_0 a_1 \alpha_1 \ldots a_n \alpha_n$$

where $\alpha_0, \alpha_1 \ldots \alpha_n \in T^*$, and

$$\mathscr{L}_G(c_p) = \lambda y_1 \ldots y_n./\alpha_0/ + y_1 + /\alpha_1/ + \cdots + y_n + /\alpha_n/.$$

Then, by induction hypothesis, there exist $\beta_1, \ldots, \beta_n \in T^*$ such that $\mathscr{L}_G(t_i) = /\beta_i/$ and $a_i \Rightarrow^* \beta_i$. This implies that

$$\mathscr{L}_G(t) = /\alpha_0/ + /\beta_1/ + /\alpha_1/ + \cdots + /\beta_n/ + /\alpha_n/,$$

and that $a \Rightarrow^* \alpha_o \beta_1 \alpha_1 \ldots \beta_n \alpha_n$. □

7. Composition as Second-Order Substitution

In order to adapt the construction of the previous section to the case of linear context-free tree grammars, we will interpret the atomic types of the abstract vocabulary as second-order types over strings.

Let $G = \langle N, T, P, s\rangle$ be a linear context-free tree grammar, and let $p \in P$ be a production rule $a(x_1, \ldots, x_n) \to t$ whose skeleton is $\langle a, a_1 \ldots a_m\rangle$. The linear λ-term [[p]] is defined to be:

$$\lambda y_1 \ldots y_m \,.\, \lambda x_1 \ldots x_n . |t|$$

where $|t|$ is inductively defined as follows:

1. $|x_i| = x_i$;
2. $|f| = /f/$, if f is a terminal symbol of rank 0;
3. $|a_i| = y_i$, if the non-terminal a_i is of rank 0;
4. $|f(t_1, \ldots, t_k)| = |t_1| + \cdots + |t_k|$, if f is a terminal symbol;
5. $|a_i(t_1, \ldots, t_k)| = y_i|t_1| \cdots |t_n|$.

Adapting Definition 7 to the case of linear context-free tree grammar is then straightforward.

DEFINITION 8. Let $G = \langle N, T, P, s\rangle$ be a linear context-free tree grammar. The Abstract Categorial Grammar $\mathscr{G}_G = \langle \Sigma_G, \Sigma_T, \mathscr{L}_G, s\rangle$ is defined as follows:

1. the abstract vocabulary Σ_G is constructed according to Definition 6;
2. the object vocabulary Σ_T is constructed according to Definition 5;
3. the lexicon $\mathscr{L}_G : \Sigma_G \to \Sigma_T$ is such that:

   (a) $\mathscr{L}_G(a) = \mathit{string}^{r_n(a)} \multimap \mathit{string}$, for all $a \in N$
   (b) $\mathscr{L}_G(c_p) = [\![p]\!]$, for all $p \in P$.

4. the distinguished type s is identical to the start symbol of G.

In order to establish the correctness of the above construction, we first state two technical lemmas concerning the operator $|\cdot|$ used in the definition of $[\![p]\!]$. Their proofs, which consist of simple inductions, are left to the reader.

LEMMA 1. *Let $G = \langle N, T, P, s\rangle$ be a linear context-free tree grammar. For all terminal trees $t \in T_T$, $|t| = /\bar{t}/$.*

LEMMA 2. *Let $G = \langle N, T, P, s\rangle$ be a linear context-free tree grammar. Let $u, u_1, \ldots, u_n \in T_{N\cup T}$, $c \in C_{N\cup T}(1)$, $a_1, \ldots, a_m \in N$, and $t \in C_T(n)$ be such that:*

1. *$a_1, \ldots, a_m$ is the sequence of occurences of non-terminal symbols in u;*
2. *$r_N(a_1) = n$;*
3. *$u = c[a_1(u_1, \ldots, u_n)]$;*

Then, $(\lambda y_1 \ldots y_m.|u|)(\lambda x_1 \ldots x_n.|t|) = \lambda y_2 \ldots y_m.|c[t[u_1, \ldots, u_n]]|$.

PROPOSITION 3. *Let $G = \langle N, T, P, s\rangle$ be a linear context-free tree grammar, and let $\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s\rangle$ be the Abstract Categorial Grammar constructed from G according to Definition 8.*

For all $a \in N$ such that $r_N(a) = n$, all $v \in C_T(n)$, and all $u_1, \ldots, u_n \in T_T$, if $a(u_1, \ldots, u_n) \Rightarrow^* v[u_1, \ldots, u_n]$ then there exists a closed λ-term $t \in \Lambda(\Sigma_G)$ such that $\vdash_{\Sigma_G} t : a$ and $\mathcal{L}_G(t)/\overline{u_1}/ \cdots /\overline{u_n}/ = /\overline{v[u_1, \ldots, u_n]}/$.

Proof. We proceed by induction on the length of the derivation $a(u_1, \ldots, u_n) \Rightarrow^* v[u_1, \ldots, u_n]$.

If $a(u_1, \ldots, u_n) \Rightarrow^* v[u_1, \ldots, u_n]$ because of a production rule $a(x_1, \ldots, x_n) \rightarrow v[x_1, \ldots, x_n]$, there exists an abstract constant corresponding to this rule, and we are done by taking t to be this abstract constant.

Now suppose that the first rule of the derivation is the production rule $p, a(x_1, \ldots, x_n) \rightarrow w[x_1, \ldots, x_n]$, whose skeleton is $\langle a, a_1 \ldots a_m\rangle$. Then, for all a_i there exist $c \in C_{N\cup T}(1)$, $w_{i1}, \ldots, w_{ir_N(a_i)} \in T_{N\cup T}$, $c' \in C_T(1)$, $w'_{i1}, \ldots, w'_{ir_N(a_i)} \in T_T$, and $v_i \in C_T(r_N(a_i))$ such that

1. $w(u_1, \ldots, u_n) = c[a_i(w_{i1}, \ldots, w_{ir_N(a_i)})]$;
2. $v(u_1, \ldots, u_n) = c'[v_i(w'_{i1}, \ldots, w'_{ir_N(a_i)})]$;
3. $c[s] \Rightarrow^* c'[s]$, for all $s \in T_{N\cup T}$;
4. $a_i(s_1, \ldots, s_{r_N(a_i)}) \Rightarrow^* v_i(s_1, \ldots, s_{r_N(a_i)})$, for all $s_1, \ldots, s_{r_N(a_i)} \in T_{N\cup T}$;
5. $w_{ij} \Rightarrow^* w'_{ij}$.

Therefore, by induction hypothesis, there exist closed λ-terms $t_1, \ldots, t_m \in \Lambda(\Sigma_G)$ such that $\vdash_{\Sigma_G} t_i : a_i$ and

$$\mathcal{L}_G(t_i)/\overline{s_1}/ \ldots /\overline{s_{r_N(a_i)}}/ = /\overline{v_i[s_1, \ldots, s_{r_N(a_i)}]}/$$

for all $s_1, \ldots, s_{r_N(a_i)} \in T_{N\cup T}$. This implies that

$$\mathcal{L}_G(t_i) = \lambda x_1 \ldots x_{r_N(a_i)}./\overline{v_i}/.$$

Then we take

$$t = c_p t_1 \cdots t_m,$$

and the result follows by iterating Lemma 2. □

PROPOSITION 4. *Let* $\mathscr{G}_G = \langle \Sigma_G, \Sigma_T, \mathscr{L}_G, s\rangle$ *be the Abstract Categorial Grammar constructed from a given context-free tree grammar* $G = \langle N, T, P, s\rangle$, *according to Definition 8.*

For all $a \in N$ *such that* $r_N(a) = n$, *and all closed* λ*-term* $t \in \Lambda(\Sigma_G)$ *such that* $\vdash_{\Sigma_G} t : a$, *there exists a context* $v \in C_T(n)$ *such that, for all* $u_1, \ldots, u_n \in T_T$, $\mathscr{L}_G(t)/\overline{u_1}/\cdots/\overline{u_n}/ = /\overline{v[u_1, \ldots, u_n]}/$, *and* $a(u_1, \ldots, u_n) \Rightarrow^* v[u_1, \ldots, u_n]$.

Proof. We proceed by induction on the structure of *t*.

If *t* is a constant then $t = c_p$ for some $p \in P$ whose skeleton is $\langle a, \epsilon\rangle$. Consequently, *p* must be of the form $a(x_1, \ldots, x_n) \rightarrow w[x_1, \ldots, x_n]$ with $w \in C_T(n)$. On the one hand, we have that

$$a(u_1, \ldots, u_n) \Rightarrow w[u_1, \ldots, u_n].$$

On the other hand, by Lemma 1,

$$\mathscr{L}_G(c_p) = \lambda x_1 \ldots x_n./\overline{w}/,$$

which implies that

$$\mathscr{L}_G(c_p)/\overline{u_1}/\cdots/\overline{u_n}/ = /\overline{w[u_1, \ldots, u_n]}/.$$

If *t* is an application then $t = c_p t_1 \ldots t_m$ for some $p \in P$ whose skeleton is $\langle a, a_1 \ldots a_m\rangle$, and each λ-term t_i must be a closed λ-term of type a_i. Consequently, by induction hypothesis, there exist contexts $v_1, \ldots, v_m$ such that $v_i \in C_T(r_N(a_i))$ and, for all all $u_1, \ldots, u_{r_N(a_i)} \in T_T$,

$$\mathscr{L}_G(t_i)/\overline{u_1}/\cdots/\overline{u_{r_N(a_i)}}/ = /\overline{v_i[u_1, \ldots, u_{r_N(a_i)}]}/,$$

and

$$a(u_1, \ldots, u_{r_N(a_i)}) \Rightarrow^* v_i[u_1, \ldots, u_{r_N(a_i)}].$$

This implies that

$$\mathscr{L}_G(t_i) = \lambda x_1 \ldots x_{r_N(a_i)}.v_i[x_1, \ldots, x_{r_N(a_i)}],$$

and the result follows by iterating Lemma 2 and applying Lemma 1. □

8. Composition as Third-Order Substitution

Finally, in this section, we define the ACG corresponding to a linear context-free rewriting system. To this end, we interpret the atomic types of the abstract vocabulary as third-order types over strings.

Let $G = \langle N, T, P, s\rangle$ be a linear context-free rewriting system, and let $p \in P$ be a production rule $\langle f, a \rightarrow a_1 a_2 \dots a_1\rangle$, whose linear transform obeys the following equation:

$$f\langle x_1, \dots, x_m\rangle = \langle \alpha_{10}x_{11}\alpha_{11}\dots x_{1_{p1}}\alpha_{1_{p1}}, \dots, \alpha_{n0}x_{n1}\alpha_{n1}\dots x_{np_n}\alpha_{np_n}\rangle.$$

We define the λ-terms $u_1, \dots, u_n$ as follows:

$$u_i = /\alpha_{i0}/ + x_{i1} + /\alpha_{i1}/ + \cdots + x_{ip_i} + /\alpha_{ip_i}/$$

The linear λ-term [[p]] is then defined to be:

$$\lambda y_1 y_2 \dots y_1.\lambda z.y_1(\lambda \mathbf{x}_1.y_2(\lambda \mathbf{x}_2.\cdots y_l(\lambda \mathbf{x}_l.zu_1\cdots u_n)))$$

Where $\mathbf{x}_1$ is the sequence of λ-variables $x_1 \dots x_{r_N(a_1)}$, $\mathbf{x}_2$ is the sequence of λ-variables $x_{r_N(a_1)+1}\dots x_{r_N(a_1)+r_N(a_2)}$, etc.

Then, the ACG corresponding to a given linear context-free rewriting system is defined as follows.

DEFINITION 9. Let $G = \langle N, T, P, s\rangle$ be a linear context-free rewriting system. The Abstract Categorial Grammar $\mathscr{G}_G = \langle \Sigma_G, \Sigma_T, \mathscr{L}_G, s\rangle$ is defined as follows:

1. the abstract vocabulary Σ_G is constructed according to Definition 6;
2. the object vocabulary Σ_T is constructed according to Definition 5;
3. the lexicon $\mathscr{L}_G : \Sigma_G \rightarrow \Sigma_T$ is such that:

   (a) $\mathscr{L}_G(a) = (string^{r_N(a)} \multimap string) \multimap string$, for all $a \in N$
   (b) $\mathscr{L}_G(c_p) = [\![p]\!]$, for all $p \in P$;

4. the distinguished type s is identical to the start symbol of G.

In order to prove the correctness of the above construction, we start by stating a technical lemma, whose proof is left to the reader.

LEMMA 3. *Let* $G = \langle N, T, P, s\rangle$ *be a linear context-free rewriting system, let* $p \in P$ *be the production rule* $\langle f, a \rightarrow a_1 \dots a_n\rangle$, *and let* $\alpha_{i1}, \dots, \alpha_{ir_N(a_i)} \in T^*$, *for* $1 \le i \le n$. *Then, there exists* $\alpha_1, \dots, \alpha_{r_N(a)} \in T^*$ *such that*

1. $f\langle\alpha_{11},\ldots,\alpha_{1r_N(a_1)},\ldots,\alpha_{n1},\ldots,\alpha_{nr_N(a_n)}\rangle = \langle\alpha_1,\ldots,\alpha_{r_N(a)}\rangle$
2. $[\![p]\!](\lambda z.z/\alpha_{11}/\cdots/\alpha_{1r_N(a_1)}/)\cdots(\lambda z.z/\alpha_{n1}/\cdots/\alpha_{nr_N(a_n)}/) = \lambda z.z/\alpha_1/\cdots$ $/\alpha_{r_N(a)}/$.

PROPOSITION 5. *Let $G = \langle N, T, P, s\rangle$ be a linear context-free rewriting system, and let $\mathcal{G}_G = \langle\Sigma_G, \Sigma_T, \mathcal{L}_G, s\rangle$ be the Abstract Categorial Grammar constructed from G according to Definition 9.*

For all $a \in N$ such that $r_N(a) = n$, and all $\alpha_1,\ldots,\alpha_n \in T^*$, if $\langle\alpha_1,\ldots,\alpha_n\rangle \in L(a)$ then there exists a closed λ-term $t \in \Lambda(\Sigma_G)$ such that $\vdash_{\Sigma_G} t : a$ and $\mathcal{L}_G(t) = \lambda z.z/\alpha_1/\cdots/\alpha_n/$.

Proof. We proceed by induction on the definition of $L(a)$.

If $f\langle\rangle = \langle\alpha_1,\ldots,\alpha_n\rangle \in L(a)$ because there exists a production rule p of the form $\langle f, a \to \epsilon\rangle$, there exists an abstract constant c_p of type a such that $\mathcal{L}_G(c_p) = \lambda z.z/\alpha_1/\cdots/\alpha_n/$.

Now suppose that $\langle\alpha_1,\ldots,\alpha_n\rangle \in L(a)$ because there exists a production rule $\langle f, a \to a_1\cdots a_m\rangle$, together with tuples $\langle\alpha_{i1},\ldots,\alpha_{ir_N(a_i)}\rangle \in L(a_i)$ such that

$$f\langle\alpha_{11},\ldots,\alpha_{1r_N(a_l)},\ldots,\alpha_{m1},\ldots,\alpha_{mr_N(a_m)}\rangle = \langle\alpha_1,\ldots,\alpha_n\rangle.$$

Hence, by induction hypothesis, there exist closed λ-terms $t_1,\ldots,t_m \in \Lambda(\Sigma_G)$ such that $\vdash_{\Sigma_G} t_i : a_i$ and $\mathcal{L}_G(t_i) = \lambda z.z/\alpha_{i1}/\cdots/\alpha_{ir_N(a_i)}/$. Then, the result follows by Lemma 3. □

PROPOSITION 6. *Let $\mathcal{G}_G = \langle\Sigma_G, \Sigma_T, \mathcal{L}_G, s\rangle$ be the Abstract Categorial Grammar constructed from a given linear context-free rewriting system $G = \langle N, T, P, s\rangle$, according to Definition 9.*

For all $a \in N$ such that $r_N(a) = n$, and all closed λ-term $t \in \Lambda(\Sigma_G)$ such that $\vdash_{\Sigma_G} t : a$, there exist $\alpha_1,\ldots,\alpha_n \in T^*$ such that $\mathcal{L}_G(t) = \lambda z.z/\alpha_1/\cdots/\alpha_n/$, and $\langle\alpha_1,\ldots,\alpha_n\rangle \in L(a)$.

Proof. We proceed by induction on the structure of *t*.

If *t* is a constant then $t = c_p$ for some $p \in P$ whose skeleton is $\langle a, \epsilon\rangle$. Consequently, p must be of the form $\langle f, a \to \epsilon\rangle$ Then, there exist $\alpha_1,\ldots,\alpha_n \in T^*$ such that $f\langle\rangle = \langle\alpha_1,\ldots,\alpha_n\rangle$. Hence, by definition, $\mathcal{L}_G(c_p) = \lambda z.z/\alpha_1/\cdots/\alpha_n/$, and $\langle\alpha_1,\ldots,\alpha_n\rangle \in L(a)$.

If t is an application then $t = c_p t_1 \ldots t_m$ for some $p \in P$ whose form is $\langle f, a \to a_1 \ldots a_m\rangle$, and each λ-term t_i must be a closed λ-term of type a_i. Therefore, by induction hypothesis, there exist $\alpha_{i1},\ldots,\alpha_{ir_N(a_i)} \in T^*$ such that $\mathcal{L}_G(t_i) = \lambda z.z/\alpha_{i1}/\cdots/\alpha_{ir_N(a_i)}/$ and $\langle\alpha_{i1},\ldots,\alpha_{ir_N(a_i)}\rangle \in L(a_i)$. Then, the result follows by Lemma 3. □

Observe that we do not have that $\alpha \in L(s)$ if and only if $/\alpha/ \in \mathcal{O}(\mathcal{G}_G)$ We have instead that $\alpha \in L(s)$ if and only if $\lambda z.z/\alpha/ \in \mathcal{O}(\mathcal{G}_G)$. This possible defect can be easily fixed by changing the distinguished type of the grammar to be a new abstract atomic type s', and by adding a new abstract constant c of type $s \multimap s'$. The lexicon is then extended in such a way that $\mathscr{L}_G(s') = string$ and $\mathscr{L}_G(c) = \lambda y.y(\lambda x.x)$.

9. Conclusions

The embeding of context-free string grammars, linear context-free tree grammars, and linear context-free rewriting systems in Abstract Categorial Grammars exemplifies some of the features of the ACG framework.

The fact that an ACG generates two languages offer an explicit control of the parse structure of the grammar. Consequently, the three encodings we have given are in fact strong equivalences.[1]

The fact that the basic objects manipulated by an ACG are linear λ-terms allows higher-order operations to be defined. Typically, tree-adjunction is such a higher-order operation (Abrusci et al., 1999; Joshi and Kulick, 1997; Mönnich, 1997), and we have seen that the possibility of defining such higher-order operations is the keystone in encoding linear context-free tree grammars and linear context-free rewriting systems.

Finally, the fact that the embeddings of the three context-free formalisms are based, respectively, on first-order, second-order, and third-order interpretations suggests the existence of an Abstract Categorial Hierachy that would allow the expressive power of the ACGs to be controlled.

Note

[1] In the case of linear context-free tree grammars, this claim might be discussed because our encoding does not give access to the tree-language generated by the linear context-free tree grammar. This possible problem may be fixed by defining the embedding in two stages as is done in (de Groote, 2002)

References

Abrusci, M., Fouqueré, C., and Vauzeilles, J., 1999, "Tree-adjoining grammars in a fragment of the Lambek calculus," *Computational Linguistics* **25**, 209–236.

Barendregt, H., 1984, *The Lambda Calculus, its Syntax and Semantics*, North-Holland, revised edition.

Carpenter, B., 1996, *Type-Logical Semantics*, Cambridge, MA and London, U.K.: MIT Press.

de Groote, P., 2001, "Towards Abstract Categorial Grammars," pp. 148–155 in *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference.*

de Groote, P., 2002, "Tree-adjoining grammars as Abstract Categorial Grammars," in pp. 145–150 *TAG+6, Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Frameworks.*

Girard, J.-Y., 1987, "Linear logic," *Theoretical Computer Science* **50**, 1–102.

Joshi, A.K. and Kulick, S., 1997, "Partial proof trees as building blocks for a categorial grammar," *Linguistic & Philosophy* **20,** 637–667.

Joshi, A.K. and Schabes, Y., 1997, "Tree-adjoining grammars," in *Handbook of Formal Languages*, Vol. 3, G.R. and A. Salomaa, ed., Springer, Chapter 2.

Mönnich, U., 1997, "Adjunction as substitution," pp. 169–178. in G.-J. Kruiff, G. Morrill, and D. Oehrle (eds.) *Formal Grammar*.

Moortgat, M., 1997, "Categorial type logics," in *Handbook of Logic and Language*, J. van Benthem and A. ter Meulen, eds., Elsevier, Chapter 2.

Morrill, G., 1994, *Type Logical Grammar: Categorial Logic of Signs*, Dordrecht: Kluwer Academic Publishers.

Oehrle, R.T., 1994, "Term-labeled categorial type systems," *Linguistic & Philosophy* **17**, 633–678.

Pollard, C., 1984, "Generalized phrase structure grammars, head grammars, and natural language," Ph.D. Thesis, Stanford University, CA.

Ranta, A., 2002, "Grammatical Framework," *Journal of Functional Programming* **14,** 145–189.

Seki, H., Matsumura, T., Fujii, M., and Kasami T., 1991, "On multiple context-free grammars," *Theoretical Computer Science* **223**, 87–120.

Vijay-Shanker, K., Weir, D. J., and Joshi, A. K. 1987, 'Characterizing structural descriptions produced by various grammatical formalisms," pp. 104–111 in *Proceedings of the 25th ACL*, Stanford, CA.

Weir, D.J., 1988, "Characterizing mildly context-sensitive grammar formalisms," Ph.D. Thesis, University of Pennsylvania.