# On Pregroups, Freedom, and (Virtual) Conceptual Necessity

Gregory M. Kobele and Marcus Kracht

**Abstract**

Pregroups were introduced in (Lambek, 1999), and provide a foundation for a particularly simple syntactic calculus. Buszkowski (2001) showed that free pregroup grammars generate exactly the $\epsilon$-free context-free languages. Here we characterize the class of languages generable by all pregroups, which will be shown to be the entire class of recursively enumerable languages. To show this result, we rely on the well-known representation of recursively enumerable languages as the homomorphic image of the intersection of two context-free languages (Ginsburg et al., 1967). We define an operation of cross-product over grammars (so-called because of its behaviour on the types), and show that the cross-product of any two free-pregroup grammars generates exactly the intersection of their respective languages. The representation theorem applies once we show that allowing 'empty categories' (i.e. lexical items without overt phonological content) allows us to mimic the effects of any string homomorphism.

## 1 Introduction

The advent of the Minimalist Program (Chomsky, 1995) heralded a shift of emphasis from the by now familiar goals of descriptive and explanatory adequacy, to a characterization of human language in terms of departures from a minimally necessary system meeting various boundary ('bare output') conditions. This involves determining both what the appropriate boundary conditions are, and what the simplest systems are that meet these conditions. Work within the minimalist program can be viewed as concerning itself primarily with the first question: the nature of the interfaces. Work in theoretical computational linguistics, on the other hand, is easily understood as seeking answers to the second. Viewing languages as sets of strings, although an obvious simplification, allows us to formulate precisely simple but interesting boundary conditions, which in turn enables feasible inquiry into the various possible formalisms which meet these boundary conditions. Perhaps the best studied such boundary condition is that the grammar formalism be able to describe a particular class $\mathcal{L}$ of languages, where $\mathcal{L}$ usually ranges over classes of languages as given by the Chomsky hierarchy (Chomsky, 1956), or various refinements thereof.[1] Results obtained here bear directly on the questions

[1]This kind of boundary condition can be understood as a first approximation of some confluence of conditions that give rise to the patterns that we actually observe

raised by Chomsky, as they can be understood as exhibiting the minimal properties a grammar formalism must have in order to be able to describe (as such) the complex of patterns attested in natural language. Properties of motivated linguistic formalisms can be compared against these minimal properties, and differences in complexity can either be interpreted as providing evidence for heretofore unacknowledged boundary conditions, or, if there is no structure in the deviations from minimality, as departures of our language faculty from perfection.

There is a broad consensus that languages are usefully described in terms of a finite set of generators (the lexicon) which are acted upon by another finite set of structure building functions. It is common to take these structure building operations to be invariant across languages, isolating the lexicon as the locus of linguistic variation.[2] Setting aside meanings, a linguistic expression consists of an exponent (here taken to be a string over a finite alphabet), and a syntactic category reflecting its distribution (which is in many theories represented as a tree). Our structure building operations, therefore, need to specify both how to combine exponents, as well as how to combine syntactic categories. In many linguistic theories, such as Tree Adjoining Grammar (TAGs, Joshi, 1987), Minimalist Grammars (MGs, Stabler and Keenan, 2003), Head Grammars (Pollard, 1984), etc, use is made of complex wrapping operations on strings when combining exponents, while in pregroup grammars, like categorial grammars, simple string concatenation is all that is available. Similarly, with respect to operations on categories, pregroup and categorial grammars make do with a simple merge operation (along with equally simple reduction (i.e. feature checking) steps), whereas TAGs and MGs have much more complicated syntactic operations. However, although simpler, neither categorial grammars nor pregroup grammars satisfy what work in computational linguistics (see esp. Shieber, 1985) has established as the appropriate boundary condition – that the formalism in question be able to describe so-called *mildly context-sensitive languages* (Joshi, 1985) – both categorial grammars (both the AB- and the Lambek calculi) and free pregroup grammars are only context-free (Bar-Hillel et al., 1960; Pentus, 1993; Buszkowski, 2001). In order to meet this and other relevant boundary conditions, categorial grammar-

(i.e. regardless of why these patterns are “actually” there). The primary motivation for adopting it as opposed to some other, more “natural”, condition (such as constraints on the pairings of sound and meaning), is that in this case it is relatively clear what questions to ask, and how to proceed.

[2]Recent work by Guillaumin (2005) raises the intriguing possibility that restricting variation to the lexicon might actually allow for simpler descriptions of the same phenomena.

ians have augmented their theories with new operations (see e.g. Steedman, 2000; Moortgat, 1996), which increase the complexity of the formalisms. In this paper we show that the pregroup grammar formalism can be made to satisfy the relevant boundary conditions *while preserving the original simplicity of the theory*. In so doing, we are offering a yardstick of (virtual) conceptual necessity against which other theories can be measured and departures therefrom noticed. This is an essential step if we wish to understand which properties of language are necessary design consequences, and which require a different explanation.

The remainder of this paper is structured as follows. In § 2, we (re)acquaint the reader with various useful definitions and abbreviatory conventions we will use in the paper, and introduce the pregroup grammar formalism. In § 3, we show that removing the requirement that the pregroups from which categories are drawn be free gives us the power to describe any recursive set of strings.

## 2 Pregroups and Grammar

Let's begin with a quick review of some basic concepts and abbreviations used in this paper. $\mathbb{N} := \{0, 1, 2, 3, \ldots\}$ is the set of natural numbers. For a finite set $A$, $\|A\|$ denotes its cardinality. The unique set of cardinality $0$ is denoted $\emptyset$. Given two sets $A$ and $B$, their cross-product $A \times B := \{\langle a, b\rangle : a \in A \wedge b \in B\}$ is the set of ordered pairs whose first component is in $A$ and whose second is in $B$. A relation $R$ over $A$ is a subset of $A \times A$. It is reflexive just in case $xRx$ for every $x \in A$. It is antisymmetric iff for every $a, b \in A$, if both $aRb$ and $bRa$, then $a = b$ and it is transitive when for every $a, b, c \in A$, if $aRb$ and $bRc$ then $aRc$. A relation $R$ is a **partial order** iff it is reflexive, antisymmetric and transitive. Given a set $A$, a string over $A$ is a finite sequence of elements $x = x_1 \ldots x_n$, $x_i \in A$ for $1 \leq i \leq n$. If $n = 0$ then $x$ is the empty string and is denoted $\epsilon$. The length of a string $x$ is denoted $\|x\|$. In particular, $\|\epsilon\| = 0$. The concatenation of two strings $x$ and $y$ is denoted by their juxtaposition $xy$, or sometimes as $x^\frown y$. If $A$ and $B$ are sets of strings, then $A_\epsilon := A \cup \{\epsilon\}$ denotes the set differing from $A$ at most in the presence of the empty string, and $AB := \{xy : x \in A$ and $y \in B\}$ is the set of strings gotten by concatenating a string in $A$ with a string in $B$. We set $A^0 := \{\epsilon\}$ and define $A^{n+1} := A^n A$. $A^n$ is the $n$-fold iteration of strings in $A$. We define $A^* := \bigcup_{n=0}^{\infty} A^n$, and $A^+ := \bigcup_{n=1}^{\infty}$. If $L \subseteq A^*$ then $L$ is called a language over $A$.

If $f : X \to Y$ is a function and $U \subseteq X$ then we write $f[U] := \{f(x) : x \in U\}$ for the image of $U$ under $f$. A **signature** $\Omega$ over a set $F$ of function symbols is a function $\Omega : F \to \mathbb{N}$. An $\Omega$–**algebra** is a pair $\mathfrak{B} = \langle B, \mathfrak{I}\rangle$ such that $\mathfrak{I}$ assigns to each $f \in F$ a function of arity $\Omega(f)$ over $B$ (i.e. $\mathfrak{I}(f) : B^{\Omega(f)} \to B$). $\mathfrak{B}$ is **partial** if $\mathfrak{I}(f)$ may also be a partial function. We shall also write $f^{\mathfrak{B}}$ in place of $\mathfrak{I}(f)$. For example, let $F = \{1, \otimes\}$ and $\Omega(1) = 0$ and $\Omega(\otimes) = 2$. An $\Omega$–algebra is a pair $\langle B, \mathfrak{I}\rangle$ such that $\mathfrak{I}(1) : \{\epsilon\} \to B$ and $\mathfrak{I}(\otimes) : B^2 \to B$ (recall that $B^0 = \{\epsilon\}$). Thus we may also view $\mathfrak{I}(1)$ as an element of $B$ instead of a nullary function. A particular example of an $\Omega$–algebra is the algebra $\mathfrak{S}(A)$ of strings over an alphabet $A$. Here, the underlying set of $\mathfrak{S}(A)$ is the set $A^*$ of strings over $A$ and $1^{\mathfrak{S}(A)} = \epsilon$ as well as $\otimes^{\mathfrak{S}(A)} = {}^\frown$, the concatenation of strings. Notice that concatenation is associative, that is, for all strings $x$, $y$ and $z$,

$$x^\frown(y^\frown z) = (x^\frown y)^\frown z \tag{1}$$

Given two algebras $\mathfrak{B} = \langle B, \mathfrak{I}\rangle$ and $\mathfrak{C} = \langle C, \mathfrak{J}\rangle$, we put $\mathfrak{B} \times \mathfrak{C} = \langle A \times C, \mathfrak{K}\rangle$ where

$$\begin{aligned}\mathfrak{K}(f)(\langle b_1, c_1\rangle, \langle b_2, c_2\rangle, \cdots, \langle b_{\Omega(f)}, c_{\Omega(f)}\rangle) :=& \\ \langle \mathfrak{I}(f)(b_1, b_2, \cdots, b_{\Omega(f)}), \mathfrak{J}(f)(c_1, c_2, \cdots, c_{\Omega(f)})\rangle&\end{aligned} \tag{2}$$

This is undefined if any of the functions $\mathfrak{I}(f)$ or $\mathfrak{J}(f)$ is undefined on their respective arguments.

Let $\mathfrak{B}$ be an algebra and $X \subseteq B$ a set. Then the algebra generated by $X$ in $\mathfrak{B}$ is obtained as follows. First, we call a subset $M$ of $A$ **closed** if, for all $f \in F$, $f^{\mathfrak{B}}(a_1, a_2, \cdots, a_{\Omega(f)}) \in M$ whenever $a_i \in M$, for $i \leq \Omega(f)$. We let $\langle X\rangle$ be the smallest (with respect to $\subseteq$) closed set containing $M$. The algebra $\mathfrak{B}$ defines an algebra $\mathfrak{X}$ on $\langle X\rangle$ via $f^{\mathfrak{X}}(a_1, a_2, \cdots, a_{\Omega(f)}) := f^{\mathfrak{B}}(a_1, a_2, \cdots, a_{\Omega(f)})$. The left hand is defined iff the right hand side is. We can give a more concrete characterisation as follows. Say that a **term** is built from variables $X = \{x_1, x_2, \ldots\}$ using the function symbols of $F$. Terms with only the binary symbol $\otimes$ as function symbols are $x_1$, $x_2$, $x_1 \otimes x_2$, $x_1 \otimes (x_2 \otimes x_1)$, and so on. If $t(x_1, \cdots, x_n)$ is a term, and $c_i$, $1 \leq i \leq n$ are elements of the algebra, then $t(c_1, \cdots, c_n)$ denotes the result of substituting the values $c_i$ for the variables $x_i$. With this, $\langle X\rangle$ consists of all elements $t(c_1, \cdots, c_n)$, where $t(x_1, \cdots, x_n)$ is a term and for all $i \leq n$, $c_i \in X$. A term $t(x_1, \cdots, x_n)$ defines a term function $t^{\mathfrak{B}} : \langle c_1, \cdots, c_n\rangle \mapsto t(c_1, \cdots, c_n)$ on $A^n$. We shall henceforth not distinguish between the term $t$ and the term function it induces on $B$. If $f$ is a term function and $U$ a set, write $f[U] := \{f(\vec{c}) : \vec{c} \in U^n\}$. We can now also say

$$\langle X\rangle = \bigcup\{f[X] : f \text{ a term function of } \mathfrak{B}\} \tag{3}$$

## 2.1 Pregroups

Pregroups offer a simplification of the AB calculus in so far as they are associative, and so are unable to distinguish between different constituent structures. Pregroups can thus be thought of as a dependency formalism. Because of their associativity, many of the operations commonly added to the AB calculus (such as type raising, or composition) are derivable as theorems in the pregroup setting.

A pregroup (see Lambek, 2001) $P = \langle M, \cdot, 1, \leq, \ ^r, \ ^l \rangle$ is a partially ordered monoid $\langle M, \cdot, 1, \leq \rangle$ with left and right inverses satisfying the following equations:[3]

$$x^\ell \cdot x \leq 1 \leq x \cdot x^\ell \text{ and } x \cdot x^r \leq 1 \leq x^r \cdot x \tag{4}$$

Note that the product of two pregroups is itself a pregroup, where $\langle p, q \rangle \leq \langle p', q' \rangle$ iff $p \leq p'$ and $q \leq q'$.

A **free** pregroup is built up from a partially ordered set of atoms $\mathbb{T}$ by first creating iterated adjoints from atoms $a \in \mathbb{T}$

$$\ldots, a^{\ell\ell}, a^\ell, a^r, a^{rr}, \ldots$$

and then taking elements of the pregroup to be sequences consisting of atoms and iterated adjoints (the unit element in the pregroup is the empty sequence). If $p, q$ are elements of the pregroup, $p \leq q$ if it holds in virtue of the equations above given the ordering over the atoms, together with antimonotony conditions: $x \leq y$ entails that $y^r \leq x^r$ and $y^\ell \leq x^\ell$.

We define a binary operation $\otimes$ over $M \times \Sigma^*$ such that $\langle p, \sigma \rangle \otimes \langle q, \tau \rangle = \langle pq, \sigma\tau \rangle$. The associativity of $\otimes$ follows from the associativity of the multiplications of the string algebra and the pregroup. A (free) pregroup grammar is a 4-tuple $G = \langle \mathbb{I}, s, P, \Sigma \rangle$, where $P = \langle M, \cdot, 1, \leq, \ ^r, \ ^l \rangle$ is a (free) pregroup, $s \in M$ is the category of sentences, and $\mathbb{I} \subseteq_{fin} M \times \Sigma_\epsilon$ is the lexicon.

[3]A monoid $\langle M, \cdot, 1 \rangle$ consists of a set $M$ together with a distinguished element $1 \in M$ and a binary operation over $M$ satisfying the following equations:

$$1 \cdot x = x = x \cdot 1 \tag{unit}$$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z \tag{associativity}$$

A partially ordered monoid $\langle M, \cdot, 1, \leq \rangle$ is a monoid $\langle M, \cdot, 1 \rangle$ together with a partial order $\leq$ over $M$ such that for any $a, b, c, d \in M$,

$$a \leq c \text{ and } b \leq d \rightarrow a \cdot b \leq c \cdot d \tag{monotonicity}$$

In this paper $x \cdot y$ is often abbreviated $xy$.

A string $\sigma \in \Sigma^*$ is **accepted** if there is some $p \in M$ such that $p \leq s$ and $\langle p, \sigma \rangle \in \langle \mathbb{I} \rangle$. We write $L(G) = \{\sigma : \langle p, \sigma \rangle \text{ is accepted}\}$.

Intuitively, in a free pregroup an atom $a \in \mathbb{T}$ is a categorial feature (like an NP) which can be selected for by an adjoint ($a^l$ or $a^r$, depending on whether it should appear to the right or the left of the selector) (see Lambek, 2004). Movement-like dependencies can be captured by means of iterated adjoints (e.g. $a^{\ell\ell}$), which look to their right (left) for an adjoint which looks for an atom on its right (left). Finally, $a \leq b$, $a, b \in \mathbb{T}$, means that anything looking for a $b$ (i.e. that has a $b^r$ or $b^\ell$ feature) will be satisfied with an $a$.

## 3 Pregroups and Language

To show that pregroup grammars can define any recursively enumerable language, we rely on Ginsburg et al.'s theorem that every r.e. language is the homomorphic image of the intersection of two context-free languages, and on Buszkowski's theorem that every context-free language is the language of some pregroup grammar.[4]

**Theorem 1 (Ginsburg et al.)** *For every recursively enumerable language $L$, there are context-free languages $L_1$ and $L_2$, and a non-length increasing homomorphism $h$ such that $L = h[L_1 \cap L_2]$.*[5]

**Theorem 2 (Buszkowski)** *For every context-free language $L$, there is a pregroup grammar $G$ such that $L = L(G)$.*

Given theorem 2, the desired result follows from theorem 1 once we show that the operations of homomorphic image and intersection on languages can be performed at the level of grammars. Proposition 3 establishes this for non-length increasing homomorphisms, and proposition 4 shows that the language of the product of two pregroup grammars is exactly the intersection of their respective languages. This has some independent interest in domains where

[4]A string homomorphism from $\Sigma^*$ to $\Gamma^*$ is a map $h$ that satisfies

$$h(\sigma\tau) = h(\sigma)h(\tau)$$

Such a map is uniquely determined by its action on $\Sigma$. A string homomorphism $h : \Sigma^* \rightarrow \Gamma^*$ is **non-length increasing** just in case every element of $\Sigma$ is mapped to an element of $\Gamma_\epsilon$ (in which case $\|h(a)\| \leq 1$, for all $a \in \Sigma$).

[5]This formulation differs from the one in (Ginsburg et al., 1967, pg 405). Inspection of their homomorphism $h$ easily reveals it to be non-length increasing, and so their proof is valid also for this formulation of their theorem.

intersection of languages can be used to model relevant phenomena, such as in linguistics – insofar as modules are thought to be independent and operate in parallel (as in the framework of Autolexical Syntax (Sadock, 1991)) – and in computational biology.[6]

**Proposition 3** *Let $G = \langle \mathbb{I}, s, P, \Sigma\rangle$ be a pregroup grammar, and $h : \Sigma^* \to \Gamma^*$ a non-length increasing homomorphism. Then there is a pregroup grammar $G^h = \langle \mathbb{I}^h, s, P, \Gamma\rangle$ such that $L(G^h) = h[L(G)]$.*

**Proof.** We extend $h$ to a map $1 \times h : P \times \Sigma^* \to P \times \Gamma^*$ by putting $(1 \times h)(\langle p, \sigma\rangle) := \langle p, h(\sigma)\rangle$. This is a homomorphism, as is easily verified. This means that for every term $t$, and all expressions $\alpha_i = \langle p_i, \sigma_i\rangle$

$$(1 \times h)(t(\alpha_1, \cdots, \alpha_n)) = t((1 \times h)(\alpha_1), \cdots, (1 \times h)(\alpha_n)) \tag{5}$$

In turn, this means that $(1 \times h)[t[X]] = t[(1 \times h)[X]]$. Hence, $(1 \times h)[t[\mathbb{I}]] = t[\mathbb{I}^h]$. It follows that

$$\begin{aligned} \langle \mathbb{I}^h\rangle &= \bigcup\{t[\mathbb{I}^h] : t \text{ a term function}\} \\ &= \bigcup\{(1 \times h)[t[\mathbb{I}]] : t \text{ a term function}\} \\ &= (1 \times h)[\langle \mathbb{I}\rangle] \end{aligned} \tag{6}$$

Now, $\gamma \in L(G^h)$ iff there is a $p \in M$ such that $\langle p, \gamma\rangle \in \langle \mathbb{I}^h\rangle$ iff there is a $p \in M$ such that $\langle p, \gamma\rangle \in (1 \times h)[\langle \mathbb{I}\rangle]$ iff there is a $p \in M$ and a $\sigma \in \Sigma^*$ such that $\gamma = h(\sigma)$ and $\langle p, \sigma\rangle \in \langle \mathbb{I}\rangle$ iff there is $\sigma \in L(G)$ such that $h(\sigma) = \gamma$. Hence, $L(G^h) = h[L(G)]$, as promised. $\dashv$

Next we shall exhibit a general construction, namely the **product** of two grammars. This works as follows. Let $G_1 = \langle \mathbb{I}_1, s_1, P_1, \Sigma\rangle$ and $G_2 = \langle \mathbb{I}_2, s_2, P_2, \Sigma\rangle$ pregroup grammars. Put $\mathbb{I}_1 \times' \mathbb{I}_2 := \{\langle p, p', \sigma\rangle : \langle p, \sigma\rangle \in \mathbb{I}_1, \langle p', \sigma\rangle \in \mathbb{I}_2\}$. Finally, put $G_1 \times G_2 := \langle \mathbb{I}_1 \times' \mathbb{I}_2, \langle s_1, s_2\rangle, P_1 \times P_2, \Sigma\rangle$.

Suppose that there are no empty lexical type assignments (i.e. the lexicon is such that $\langle p, \sigma\rangle \in \mathbb{I}$ only if $\sigma \in \Sigma$). Then an analysis of a string of length $n$ will contain exactly $n$ occurrences of lexical elements. So, $\sigma \in \langle \mathbb{I}\rangle$ iff there is a term $t(x_1, \cdots, x_n)$ containing exactly $n-1$ (!) occurrences of $\otimes$ and lexical elements $\alpha_i = \langle p_i, \sigma_i\rangle$, $1 \leq i \leq n$, such that

$$t(\alpha_1, \cdots, \alpha_n) = \langle p, \sigma\rangle \tag{7}$$

[6]As pertains to this latter domain, Chiang (2004) argues that what he calls *weak parallelism* is inappropriate for modeling various biological phenomena, because the structural descriptions assigned by the two languages are not correlated. He proposes a solution within the framework of Tree Adjoining Grammars. Our operation of product over pregroup grammars also has the properties that he requires.

for some $p$. If $\otimes$ is associative, we can choose the following term:

$$(\cdots((\alpha_1 \otimes \alpha_2) \otimes \alpha_2) \cdots \alpha_n) \tag{8}$$

This will be useful for the next theorem.

**Proposition 4** *For $G_1, G_2$ pregroup grammars (without empty lexical type assignments), $L(G_1 \times G_2) = L(G_1) \cap L(G_2)$.*

**Proof.** Define the following maps. $\pi_1 : P_1 \times P_2 \times \Sigma^* \to P_1 \times \Sigma^* : \langle p, p', \sigma\rangle \mapsto \langle p, \sigma\rangle$, and $\pi_2 : P_1 \times P_2 \times \Sigma^* \to P_1 \times \Sigma^* : \langle p, p', \sigma\rangle \mapsto \langle p', \sigma\rangle$. These maps are actually homomorphisms. Furthermore, $\pi_1[\mathbb{I}_1 \times' \mathbb{I}_2] = \mathbb{I}_1$ as well as $\pi_2[\mathbb{I}_1 \times' \mathbb{I}_2] = \mathbb{I}_2$. From this we can already deduce that if $\sigma \in L(G_1 \times G_2)$ then $\sigma \in L(G_1) \cap L(G_2)$. For if $\sigma \in L(G_1 \times G_2)$ then there are $p$, $p'$ such that $p \leq s_1$ and $p' \leq s_2$ and $\langle p, p', \sigma\rangle \in \langle \mathbb{I}_1 \times' \mathbb{I}_2\rangle$, then $\langle p, \sigma\rangle = \pi_1(\langle p, p', \sigma\rangle \in \pi_1[\langle \mathbb{I}_1 \times' \mathbb{I}_2\rangle] = \langle \pi_1[\mathbb{I}_1 \times' \mathbb{I}_2]\rangle = \langle \mathbb{I}_1\rangle$. Similarly $\langle p', \sigma\rangle \in \langle \mathbb{I}_2\rangle$ is established. For the converse we need to make use of our further assumptions. Suppose that $\sigma \in L(G_1)$ and $\sigma \in L(G_2)$. Then there is a term $t(y_1, \cdots, y_j)$ and elements $\alpha_i \in \mathbb{I}_1$, such that $t(\alpha_1, \cdots, \alpha_j) = \langle p, \sigma\rangle$ for some $p \leq s_1$; and there is a term $t'(z_1, \cdots, z_k)$ and elements $\alpha'_i \in \mathbb{I}_2$ such that $t(\alpha'_1, \cdots, \alpha'_k) = \langle p', \sigma\rangle$ for some $p' \leq s_2$. We are not guaranteed that $t$ and $t'$ are the same term. However, under the assumptions made, as the discussion above has revealed, we do have $j = k$, and we can use the same term. Moreover, we have $\alpha_i = \langle p_i, \sigma_i\rangle$ and $\alpha'_i = \langle p'_i, \sigma_i\rangle$ for certain $p_i \in M_1$ and $p'_i \in M_2$. It follows that

$$t(\langle p_1, p'_1, \sigma_1\rangle, \cdots, \langle p_j, p'_j, \sigma_j\rangle) = \langle p, p', \sigma\rangle \tag{9}$$

and since $\langle p, p'\rangle \leq \langle s_1, s_2\rangle$, we now have $\sigma \in L(G_1 \times G_2)$. $\dashv$

The theorem can be improved. It is often customary to allow for the empty string $\epsilon$ in the lexicon. In this case, the product grammar shall contain also the following items: $\langle p, 1, \epsilon\rangle$ iff $\langle p, \epsilon\rangle \in \mathbb{I}_1$ and $\langle 1, p', \epsilon\rangle$ iff $\langle p', \epsilon\rangle \in \mathbb{I}_2$. Or, equivalently, we assume that both lexica contain the entry $\langle 1, \epsilon\rangle$. Intuitively, this is so because, unlike the actual letters of $\Sigma$, which both grammars must recognize, one grammar may ‘see’ an empty string without the other one being required to.

We are now able to show our main theorem:

**Theorem 5** *For every recursively enumerable $L$, there is a pregroup $G$ such that $L = L(G)$.*

**Proof.** By theorem 1, there are context-free languages $L_1$ and $L_2$, and a string homomorphism $h$ such that $L = h[L_1 \cap L_2]$. By theorem 2, there are pregroup grammars $G_1$ and $G_2$ such that $L_i = L(G_i)$, for $i \in \{1, 2\}$. The theorem follows from propositions 3 and 4 by taking $G = (G_1 \times G_2)^h$. $\dashv$

## References

Bar-Hillel, Y., C. Gaifman, and E. Shamir. 1960. On categorial and phrase-structure grammars. *Bulletin of the research council of Israel* F:1–16.

Buszkowski, W. 2001. Lambek grammars based on pregroups. In *Logical aspects of computational linguistics*, ed. P. de Groote, G. Morrill, and C. Retoré, volume 2099 of *Lecture Notes in Artificial Intelligence*. New York: Springer.

Chiang, D. 2004. Uses and abuses of intersected languages. In *Proceedings of the 7th International Conference on Tree Adjoining Grammar and Related Formalisms*, 9–15. Vancouver, BC. Canada.

Chomsky, N. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2:113–124.

Chomsky, N. 1995. *The minimalist program*. Cambridge, Massachusetts: MIT Press.

Chomsky, N. 2004. Beyond explanatory adequacy. In *Structures and beyond: The cartography of syntactic structures*, ed. A. Belletti, volume 3 of *Oxford Studies in Comparative Syntax*, chapter 3. Oxford University Press.

Ginsburg, S., S. A. Greibach, and M. A. Harrison. 1967. One-way stack automata. *Journal of the Association for Computing Machinery* 14:389–418.

Guillaumin, M. 2005. A note on the relative succinctness of MGs and equivalent MCFGs. Ms. UCLA.

Joshi, A. 1985. How much context-sensitivity is necessary for characterizing structural descriptions. In *Natural language processing: Theoretical, computational and psychological perspectives*, ed. D. Dowty, L. Karttunen, and A. Zwicky, 206–250. NY: Cambridge University Press.

Joshi, A. 1987. An introduction to tree adjoining grammars. In *Mathematics of language*, ed. A. Manaster-Ramer. Amsterdam: John Benjamins.

Lambek, J. 1958. The mathematics of sentence structure. *American Mathematical Monthly* 65:154–170.

Lambek, J. 1999. Type grammars revisited. In *Logical aspects of computational linguistics*, ed. A. Lecomte, F. Lamarche, and G. Perrier, volume 1582 of *Lecture Notes in Artificial Intelligence*, 1–27. New York: Springer.

Lambek, J. 2001. Type grammars as pregroups. *Grammars* 4:21–35.

Lambek, J. 2004. A computational algebraic approach to English grammar. *Syntax* 7:128–147.

Moortgat, M. 1996. Categorial type logics. In *Handbook of logic and language*, ed. J. van Benthem and A. ter Meulen. Amsterdam: Elsevier.

Pentus, M. 1993. Lambek grammars are context free. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*, 429–433. Los Alamitos, California: IEEE Computer Society Press.

Pollard, C. 1984. Generalized context-free grammars, head grammars, and natural language. Doctoral Dissertation, Stanford.

Sadock, J. M. 1991. *Autolexical syntax. a theory of parallel grammatical representations*. Studies in Contemporary Linguistics. Chicago: University of Chicago Press.

Shieber, S. M. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8:333–343.

Stabler, E. P., and E. L. Keenan. 2003. Structural similarity within and among languages. *Theoretical Computer Science* 293:345–363.

Steedman, M. 2000. *The syntactic process*. MIT Press.