Identification in the limit of categorial grammars

M. Kanazawa

Computer Science/Department of Software Technology

Report CS-R9351 July 1993

CWI is the National Research Institute for Mathematics and Computer Science. CWI is part of the Stichting Mathematisch Centrum (SMC), the Dutch foundation for promotion of mathematics and computer science and their applications.
SMC is sponsored by the Netherlands Organization for Scientific Research (NWO). CWI is a member of ERCIM, the European Research Consortium for Informatics and Mathematics.

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

Identification in the Limit of Categorial Grammars

Makoto Kanazawa
Department of Linguistics
Stanford University
Stanford, CA 94305–2150, USA
kanazawa@csli.stanford.edu

Abstract

The class of classical categorial grammars which assign at most k types to each symbol in the alphabet is shown to be identifiable in the limit from positive data in the sense of Gold (1967), for any k. A corollary to this is that the entire class of context-free languages becomes learnable from positive data if the learner is provided with the additional information that enables her to distinguish between different uses of a lexically ambiguous symbol. Our algorithm incorporates Buszkowski's algorithm (Buszkowski 1987, Buszkowski and Penn 1990), which determines a rigid classical categorial grammar (which assigns at most one type to each symbol) from data consisting of functor-argument structures.

1991 CR Subject Classification: F.4.3, I.2.6.
Keywords and Phrases: categorial grammar, context-free languages, formal language theory, identification in the limit, inductive inference, learnability.
NOTE: The author was sponsored by project NF 102/62–356 ('Structural and Semantic Parallels in Natural Languages and Programming Languages'), funded by the Netherlands Organization for the Advancement of Research (N.W.O.).

1 Introduction

Learnability is an important concept in linguistics. It is a property of a class of languages. A class of languages is learnable if there is a learning strategy that succeeds in learning an arbitrary language in the class. It is considered that the class of natural languages has to be learnable, since a normal child is capable of learning any natural language that happens to be spoken around her.

Report CS-R9351
ISSN 0169-118X
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Among the various paradigms of learning current in the computational learning theory, we consider the classical paradigm of *identification in the limit* due to Gold (1967) (see also Osherson, Stob, and Weinstein 1986). Gold's paradigm was partly motivated by first language acquisition, and identification in the limit is in fact the only mathematical notion of learning that linguists have paid serious attention to (Wexler and Culicover 1980).

In Gold's model, language learning is considered an infinite process in which the learner is presented with an infinite stream of sentences of the target language, one sentence at a time. Each time the learner receives a sentence, she makes a guess as to the identity of the target language, on the basis of a finite number of sentences she has received so far. Each guess is made in the form of a grammar supposed to account for the language. As the learner receives more and more data, she makes successive guesses, possibly different from the previous ones. Thus, corresponding to the infinite sequence of sentences presented to the learner, she produces an infinite sequence of grammars (figure 1).

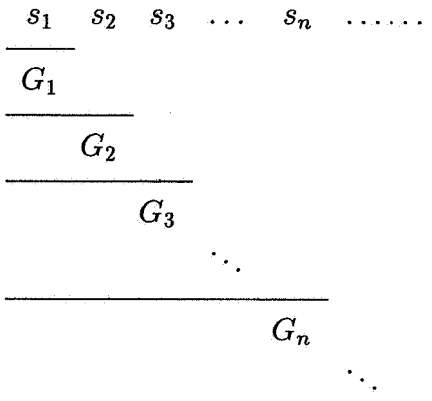


Figure 1: Language learning.

It is assumed that every sentence of the target language eventually appears in the infinite sequence of sentences and only sentences of the target language appear. Now the learning is considered successful when there is some point beyond which the learner always makes the same guess and that guess is a correct grammar for the language. Note that at any finite stage, one can never tell whether the learning has succeeded or not, since the guess might change at the next moment. Gold called this criterion of successful learning *identification in the limit*.

Under this criterion, a class of languages is learnable if there is a learning strategy that identifies in the limit any language in the class, no matter in what order the sentences of the language are presented. A class of languages learnable in this sense is called *identifiable in the limit*.

Note that in the above model, the information that the learner has about the target language at any given point consists of only *positive data* about the language (i.e., strings in the language). Gold also considered identification from complete data, which include both positive and negative data (strings not in

the target language) marked as such. Complete data makes identification much easier. In this paper, we only consider identification from positive data. (Other variations of the above model are possible, too.)

Gold (1967) studied identifiability in the limit of classes of formal languages. The initial result obtained by him about identification from positive data was extremely negative. He showed that the class of all finite languages is a maximal identifiable class. If a class contains all finite languages and at least one infinite language, then the class is not identifiable in the limit from positive data. Therefore, identification in the limit from positive data was considered extremely weak; not even the class of regular languages, which lies at the bottom of the Chomsky Hierarchy, is identifiable.

However, recent works (e.g., Angluin 1979, 1980, 1982) have shown that Gold's result was very misleading. Many interesting classes of languages have been shown to be identifiable in the limit from positive data, which cross-cut the Chomsky Hierarchy. The strongest result of this kind is due to Shinohara (1990a, 1990b): the class of languages generated by context-sensitive grammars which employ at most k rules, for any fixed k, is identifiable in the limit from positive data. Note that for any k, although this class contains infinitely many languages, it does not contain all finite languages, so Gold's result does not apply. This shows that there is a sense in which identification in the limit from positive data can be said to be quite powerful.

In this paper, we prove a result similar to Shinohara's with respect to classical categorial grammars: the class of languages that are generated by classical categorial grammars which assign at most k types to each symbol in the alphabet is identifiable in the limit from positive data, for each k. We show this by presenting a relatively concrete algorithm, which makes use of Buszkowski's algorithm (Buszkowski 1987, Buszkowski and Penn 1990) for determining a rigid categorial grammar (i.e., a categorial grammar which assigns at most one type to each symbol) from data consisting of functor-argument structures. Given Shinohara's theorem, our result is not surprising; indeed, a straightforward reduction of our result to Shinohara's is possible (see Appendix). Nevertheless, since the algorithm provided by the proof of Shinohara's theorem is of a rather abstract character,[1] our use of Buszkowski's algorithm may be of independent interest.

The fact that the class of rigid categorial grammars is identifiable in the limit from positive data (consisting of strings) also has an interesting consequence on the issue of what kind of additional information compensates for the lack of explicit negative information in positive data. Suppose that the data given to the learner is not plain strings in the target language but *disambiguated strings*, that is, strings where each occurrence of a symbol is annotated with a subscript to indicate type ambiguity. Given such data, the entire class of context-free languages becomes learnable while negative data is still precluded.

[1]Shinohara's theorem depends on a theorem by Wright (1989), which in turn depends on a theorem by Angluin (1980). Following the proofs of these theorems provides an algorithm that identifies the class in question.

2 Preliminaries

2.1 Identification in the Limit

Throughout the paper, Σ stands for a fixed finite alphabet of symbols. Σ^* denotes the set of all strings of symbols from Σ, and Σ^+ denotes the set of all non-empty strings of symbols from Σ. The term 'language' is usually used to mean a subset of Σ^*. For the purposes of this paper, however, we disregard the empty string, and consider only *ϵ-free* languages, that is, subsets of Σ^+. Thus, in this paper, we use *language* to mean ϵ-free language. Likewise, we simply say *string* to mean non-empty string.

A *grammar system* consists of a class Ω and a 'naming' function $\mathrm{L}\colon \Omega \to \mathrm{pow}(\Sigma^+)$, where Ω is understood to be a domain on which effective computation can be carried out (Ω could be the set of natural numbers, or it could be a set of some finitary objects like Turing machine programs or rewriting systems). Members of Ω are here called *grammars*. If G is a grammar, then $\mathrm{L}(G)$ is called the *language generated by* G. The naming function $\mathrm{L}(\cdot)$ usually satisfies the following two conditions: (1) the relation $s \in \mathrm{L}(G)$ between $s \in \Sigma^+$ and $G \in \Omega$ is at least semi-decidable (r.e.); (2) from any finite language $D \subseteq \Sigma^+$, a grammar $G \in \Omega$ such that $\mathrm{L}(G) = D$ can be effectively computed (so Ω contains names for all finite languages).

Let a grammar system $\langle \Omega, \mathrm{L}(\cdot) \rangle$ be given. Let $\langle s_i \rangle_{i \in \omega} = s_1, s_2, s_3, \ldots$ be an infinite sequence of strings from Σ^+. Let φ be a (partial) function that maps finite sequences of strings from Σ^+ to grammars in Ω ($\varphi\colon \bigcup_{k \geq 1} (\Sigma^+)^k \to \Omega$). Then φ is said to *converge* to G on $\langle s_i \rangle_{i \in \omega}$ iff there exists an n such that for all $m \geq n$, $\varphi(\langle s_1, \ldots, s_m \rangle)$ is defined and $\varphi(\langle s_1, \ldots, s_m \rangle) = G$. If $\mathcal{G}$ is a class of grammars from Ω, φ is said to identify $\mathcal{G}$ in the limit from positive data iff for every G in $\mathcal{G}$ and for every infinite sequence $\langle s_i \rangle_{i \in \omega}$ that enumerates $\mathrm{L}(G)$, φ converges on $\langle s_i \rangle_{i \in \omega}$ to some G' in $\mathcal{G}$ such that $\mathrm{L}(G') = \mathrm{L}(G)$. $\mathcal{G}$ is said to be *non-effectively identifiable in the limit from positive data* iff there is a function that identifies $\mathcal{G}$ in the limit from positive data. If there is an effectively computable function that identifies $\mathcal{G}$ in the limit from positive data, then $\mathcal{G}$ is said to be *identifiable in the limit from positive data*. We also say that a class $\mathcal{L}$ of languages is (non-effectively) identifiable in the limit from positive data iff there is a class $\mathcal{G}$ of grammars which is (non-effectively) identifiable in the limit from positive data such that $\mathcal{L} = \{ \mathrm{L}(G) \mid G \in \mathcal{G} \}$. Since we do not consider identification from complete data in this paper, we shall often simply say 'identify in the limit', etc., to mean 'identify in the limit from positive data', etc. Note that in general identifiability of a class of languages depends on the grammar system under consideration.

Theorem 1 (Gold) *With respect to any grammar system, if a class of languages contains one infinite language and all finite subsets of it, it is not non-effectively identifiable in the limit from positive data.*

If a grammar system satisfies condition (2) above, then the class of all finite languages is identifiable in the limit from positive data, so it is a maximal identifiable class.

2.2 Classical Categorial Grammars

We consider the grammar system $\langle \mathcal{C}, \mathrm{L}(\cdot) \rangle$ consisting of the class $\mathcal{C}$ of classical categorial grammars of Ajdukiewicz and Bar-Hillel, under the usual interpretation. In this system, a grammar G is a finite relation between Σ and Tp ($G \subset \Sigma \times \mathrm{Tp}$ and G is finite),[2] where Tp is the set of *types*. Types are constructed from *primitive types* by two type-forming operators, / and \; that is, Tp is the smallest set containing Pr such that if $A, B \in \mathrm{Tp}$, then $A\backslash B, B/A \in \mathrm{Tp}$, where Pr is the countably infinite set of primitive types. One member t of Pr is singled out as the *distinguished type*. If a grammar G relates $a \in \Sigma$ with $A \in \mathrm{Tp}$ ($\langle a, A \rangle \in G$), we say G *assigns* A *to* a and write $G\colon a \mapsto A$.

We say that a finite sequence of types $A_1, \ldots, A_n$ *cancels* to a type A_{n+1}, if the expression $A_1, \ldots, A_n \Rightarrow A_{n+1}$ is derivable from the axiom schema:

$$A \Rightarrow A$$

using the rules:

$$\frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow A\backslash B}{\Gamma, \Delta \Rightarrow B}$$

$$\frac{\Gamma \Rightarrow B/A \quad \Delta \Rightarrow A}{\Gamma, \Delta \Rightarrow B}$$

(Γ and Δ stand for finite sequences of types). Now $\mathrm{L}(G)$ is defined to be the set of strings $s = a_1 \ldots a_n$ such that for some types $A_1, \ldots, A_n$, $G\colon a_i \mapsto A_i$ ($1 \leq i \leq n$) and $A_1, \ldots, A_n$ cancels to t. It is known that $L = \mathrm{L}(G)$ for some classical categorial grammar G if and only if L is an ϵ-free context-free language, and that the question whether $s \in \mathrm{L}(G)$ is decidable in polynomial time. $\langle \mathcal{C}, \mathrm{L}(\cdot) \rangle$ satisfies the conditions (1) and (2) above on the naming function.

If G is a partial function from Σ to Tp, then G assigns at most one type to each symbol in Σ. Such a grammar is called *rigid*. If a grammar G assigns at most k types to each symbol, then G is called a *k-valued* grammar. Let $\mathcal{C}_k$ denote the class of k-valued grammars, and let $\mathcal{L}_k = \{ \mathrm{L}(G) \mid G \in \mathcal{C}_k \}$. Note that for $k \geq 2$, $\mathcal{L}_k$ is infinite, and if $|\Sigma| \geq 2$, $\mathcal{L}_1$ is, too.

We state our main result in this paper:

Theorem 2 (Main Theorem) *For each k, $\mathcal{C}_k$ (and hence $\mathcal{L}_k$) is identifiable in the limit from positive data.*

[2] In this paper, we use $\subset$ to mean proper subset.

3 Buszkowski's Algorithm

The question of learnability has not been addressed with respect to categorial grammars. There is a very important work by Buszkowski (1987), however, in which he presents a simple intuitive algorithm that computes a classical categorial grammar from linguistic data. The input to the algorithm is a finite set consisting of *functor-argument structures*, and the output is a rigid classical categorial grammar which is compatible with the input data (if there is one). The algorithm, which is similar to the one for the typing problem in lambda and combinatory calculi, is an interesting application of unification, and it has some nice formal properties, as proved by Buszkowski and Penn (1990). Although Buszkowski does not consider the question of learnability, his algorithm can in fact be used in an algorithm that identifies C_k in the limit.

3.1 Definitions of Basic Notions

Before presenting Buszkowski's algorithm, we introduce some basic notions. Henceforth, 'grammar' means 'classical categorial grammar', unless otherwise specified.

3.1.1 F-structure

A *functor-argument structure* (*F-structure* for short) (over Σ) is just a binary branching tree where leaves are labeled by symbols from Σ, and, for each internal node, one of its daughters is distinguished as the functor (the other daughter is the argument). Figure 2 is an example of F-structure.

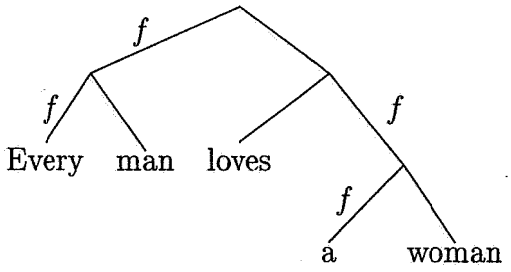


Figure 2: F-structure.

'f' here indicates functors. Symbols in Σ are identified with F-structures of depth 0. An *F-language* is just a set of F-structures.

If we view trees as terms, then F-structures can be regarded as terms built up from constants in Σ and two binary function symbols f_L and f_R (corresponding to the case where the left daughter is the functor and the case where the right daughter is the functor, respectively). For example, the F-structure in figure 2 can be written $f_L(f_L(\text{every}, \text{man}), f_R(\text{loves}, f_L(\text{a}, \text{woman})))$. Thus, F-structures

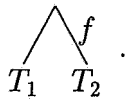
are strings over $\Sigma \cup \{f_L, f_R\}$ of a certain kind (parentheses and commas are not strictly necessary), and F-languages are certain subsets of $(\Sigma \cup \{f_L, f_R\})^+$. (The set of all F-structures over Σ is a context-free language over $\Sigma \cup \{f_L, f_R\}$.)

A *substructure* of an F-structure T is an F-structure that occurs as part of T. Each node ν of T determines a substructure, namely that portion of T that lies below ν (including ν). If we view F-structures as terms, then substructures are subterms.

The *yield* of an F-structure T, yield(T), is the string of symbols labeling the leaves of T. If we view T as a string in $(\Sigma \cup \{f_L, f_R\})^+$, then yield($T$) is the result of erasing the occurrences of f_L and f_R in T.

The type assignment to symbols by a grammar G is extended to a type assignment to F-structures in the following way.

- If G assigns type B/A to F-structure T_1 and type A to F-structure T_2, then G assigns type B to 

.
- If G assigns type A to F-structure T_1 and type $A\backslash B$ to F-structure T_2, then G assigns type B to 

.

Let us also write $G{:}\,T \mapsto A$ for 'G assigns type A to F-structure T'. So, for example, if

$$G{:}\quad \begin{array}{lll} \text{John} & \mapsto & e \\ \text{walks} & \mapsto & e\backslash t \end{array}$$

then

$$G{:}\quad \begin{array}{c} f \\ \text{John}\quad \text{walks} \end{array} \quad \mapsto \quad t,$$

but not

$$G{:}\quad \begin{array}{c} f \\ \text{John}\quad \text{walks} \end{array} \quad \mapsto \quad t.$$

Let SubTp(G) be the set of subtypes of the types in range(G). If $G{:}\,T \mapsto A$, then $A \in \text{SubTp}(G)$.

The *F-language of* G, FL(G), is defined to be $\{\, T \mid G{:}\,T \mapsto t \,\}$. L($G$) can then be defined in terms of FL(G); $\text{L}(G) = \{\, \text{yield}(T) \mid T \in \text{FL}(G) \,\}$.

A *parse* of an F-structure T by a grammar G is a function h from the set of nodes of T to Tp which satisfies the following conditions:

- h maps the root node of T to t.

- If a leaf node ν of T is labeled by a symbol a, then h maps ν to a type A such that $G: a \mapsto A$.
- If a node ν of T has a left daughter ν_1 and a right daughter ν_2, then h maps ν to B if and only if either (i) ν_1 is marked as the functor and for some type A, h maps ν_1 to B/A and ν_2 to A, or (ii) ν_2 is marked as the functor and for some type A, h maps ν_1 to A and ν_2 to $A\backslash B$.

If a parse of some F-structure T by G maps a node ν of T to A and T' is the substructure of T determined by ν, then $G: T' \mapsto A$. Moreover, if $G: T \mapsto t$, then there must be a parse of T by G. So $\mathrm{FL}(G) = \{\, T \mid G \text{ has a parse of } T \,\}$.

Note that $\langle \mathcal{C}, \mathrm{FL}(\cdot) \rangle$ can be viewed as a grammar system (over an extended alphabet).

3.1.2 Substitution

Let $\mathrm{Var} = \mathrm{Pr} - \{t\}$. We call the elements of Var *variables*. There are denumerably many variables, so we assume $\mathrm{Var} = \{x_1, x_2, x_3, \ldots\}$.

A *substitution* is a function $\sigma: \mathrm{Var} \to \mathrm{Tp}$, which is extended to a function $\sigma: \mathrm{Tp} \to \mathrm{Tp}$ as follows:

$$\begin{aligned} \sigma(t) &= t \\ \sigma(A\backslash B) &= \sigma(A)\backslash\sigma(B) \\ \sigma(B/A) &= \sigma(B)/\sigma(A) \end{aligned}$$

If σ and σ' are substitutions, σ is said to be *more general* than σ' if there is a substitution τ such that $\sigma' = \tau \circ \sigma$.

A substitution σ is said to *unify* a set of types $\{A_1, \ldots, A_n\}$ if $\sigma(A_1) = \cdots = \sigma(A_n)$. We say that σ unifies a family of sets of types, if σ unifies each set in the family. The notion of *most general unifier* is understood in the usual way: σ is a most general unifier of a family $\mathcal{S}$ of sets of types iff for every unifier σ' of $\mathcal{S}$, σ is more general than σ'. There is an algorithm that decides whether a given family of sets of types has a unifier, and if it does, computes a most general unifier of it.

$\sigma[G]$ denotes the grammar obtained by performing substitution σ in the type assignment of G. That is, $\sigma[G] = \{\, \langle a, \sigma(A) \rangle \mid \langle a, A \rangle \in G \,\}$. $\sigma[G]$ is called a *substitution instance* of G. Two grammars that are substitution instances of each other are identical for all intents and purposes and are called *alphabetic variants*. The following is easy to see:

Lemma 1 *If* $\sigma[G] \subseteq G'$, *then* $\mathrm{FL}(G) \subseteq \mathrm{FL}(G')$.

3.2 Algorithm RG

Buszkowski's algorithm RG takes a finite set of F-structures as input and returns a rigid grammar compatible with it as output, if there is one:

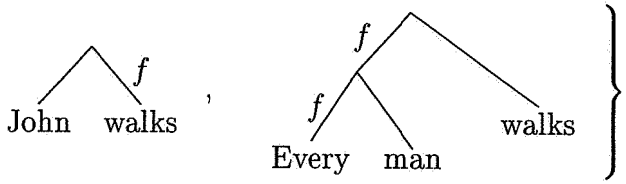
$$\mathrm{RG}: D = \{T_1, T_2, \ldots, T_n\} \longmapsto \underset{\text{rigid}}{G}$$

The algorithm essentially uses unification of sets of types.

Buszkowski's algorithm RG.

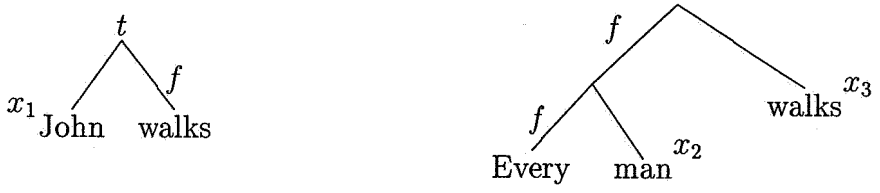
- **input:** a finite set D of F-structures.
- **output:** a rigid grammar G such that $D \subseteq \mathrm{FL}(G)$ (if there is one).

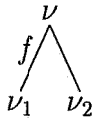
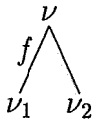
We illustrate the algorithm using the following example:

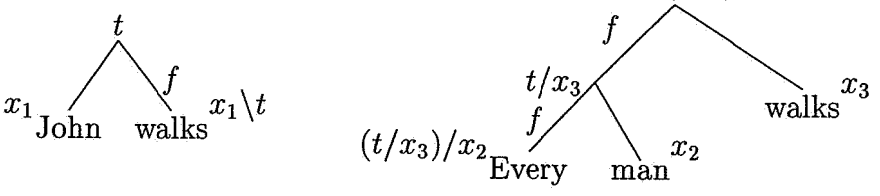
$D =$ { 

}

1. Assign a type to each node of F-structures in D as follows:
   (a) Assign t to the root node of each F-structure in D.
   (b) Assign distinct variables to argument nodes.



   (c) Compute types of functor nodes as follows:
   - If nodes ν, ν_1, ν_2 occur in the configuration  and B is assigned to ν and A is assigned to ν_2, then assign B/A to ν_1.
   - If nodes ν, ν_1, ν_2 occur in the configuration  and B is assigned to ν and A is assigned to ν_1, then assign $A\backslash B$ to ν_2.



2. For each symbol a in the alphabet, collect the set of types S_a assigned to leaf nodes labeled by a in the previous step. Obtain a grammar GF(D) (the *general form determined by* D) by letting GF(D) = $\{ \langle a, A \rangle \mid A \in S_a \}$.

$$\begin{array}{rrl} \text{GF}(D): & \text{John} & \mapsto \ x_1 \\ & \text{walks} & \mapsto \ x_1\backslash t,\ x_3 \\ & \text{every} & \mapsto \ (t/x_3)/x_2 \\ & \text{man} & \mapsto \ x_2 \end{array}$$

3. Unify the types assigned to the same symbol. Obtain a most general unifier σ of $\{ S_a \mid a \in \Sigma \}$. (RG fails if unification fails.) For the present example, σ is a most general unifier that unifies the two types assigned to *walks*.

$$\sigma(x_3) = x_1\backslash t$$

4. Let RG(D) = σ[GF(D)]. (RG(D) is the *rigid grammar determined by* D.)

$$\begin{array}{rrl} \text{RG}(D): & \text{John} & \mapsto \ x_1 \\ & \text{walks} & \mapsto \ x_1\backslash t \\ & \text{every} & \mapsto \ (t/(x_1\backslash t))/x_2 \\ & \text{man} & \mapsto \ x_2 \end{array}$$

(Note the change in the type assigned to *every*.)

3.3 Properties of RG

In this section, we state some important properties of the algorithm RG. Some of them will not be needed in the proof of the main theorem, but they may be of independent interest.

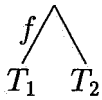
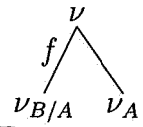
3.3.1 Basic Properties

First we present some results from Buszkowski and Penn (1990).

Lemma 2 (Buszkowski and Penn) FL(GF(D)) = D.

PROOF. The following are easy to see. (i) If B/A or $A\backslash B \in$ SubTp(GF(D)), then $A \in$ Var. (ii) For any $B \in$ SubTp(GF(D)), if $B \neq t$, step 1 of the algorithm RG assigns B to exactly one node (call it ν_B) in D. (iii) Step 1 of RG provides a parse h_i of each F-structure $T_i \in D$ by GF(D). By (iii), $D \subseteq$ GF(D). To show that GF(D) $\subseteq D$, we prove

(*) If GF(D):$T \mapsto B$, then T is the same F-structure as the substructure determined by some node ν that step 1 of RG assigns B,

by induction on the depth of T. If T is a symbol a in Σ, then $\mathrm{GF}(D)\colon a \mapsto B$, and (*) is clearly satisfied. Let $T =$  (the other case is similar). Then for some A, $\mathrm{GF}(D)\colon T_1 \mapsto B/A$ and $\mathrm{GF}(D)\colon T_2 \mapsto A$. A must be a variable, so neither B/A nor A is t. So by induction hypothesis, T_1 is the same F-structure as the substructure determined by the node $\nu_{B/A}$ in D, and T_2 is the same F-structure as the substructure determined by the node ν_A in D. In order for $\nu_{B/A}$ to be assigned B/A in step 1 of RG, $\nu_{B/A}$ and ν_A must occur in the configuration  . ν must be assigned B by step 1 of RG, and  is the same F-structure as the substructure determined by the node ν in D. ■

Lemma 3 (Buszkowski and Penn) *Let D be a finite set of F-structures. Then for any grammar G the following are equivalent:*

(i) $D \subseteq \mathrm{FL}(G)$

(ii) *there is a substitution σ such that $\sigma[\mathrm{GF}(D)] \subseteq G$.*

PROOF. (ii) $\Rightarrow$ (i) follows from Lemma 1 and Lemma 2.

(i) $\Rightarrow$ (ii). Assume $D \subseteq \mathrm{FL}(G)$. For each F-structure T_i in D, G has a parse h_i of T_i. Let $h = \bigcup_i h_i$. Define a substitution σ by putting for each variable $x_j \in \mathrm{Var} \cap \mathrm{SubTp}(\mathrm{GF}(D))$, $\sigma(x_j) = h(\nu_{x_j})$. (See the proof of Lemma 2). Then by induction on the complexity of type $A \in \mathrm{SubTp}(\mathrm{GF}(D))$, we can easily see that $\sigma(A) = h(\nu_A)$ if $A \neq t$. Therefore, if $\mathrm{GF}(D)\colon a \mapsto A$, then $G\colon a \mapsto \sigma(A)$. (The case $A = t$ is taken care of by the fact that $a \in D$.) This means $\sigma[\mathrm{GF}(D)] \subseteq G$. ■

Proposition 4 (Buszkowski and Penn) *Let D be a finite set of F-structures. Then, for any rigid grammar G, the following are equivalent:*

(i) $D \subseteq \mathrm{FL}(G)$

(ii) $\mathrm{RG}(D)$ *exists and $\tau[\mathrm{RG}(D)] \subseteq G$ for some substitution τ.*

PROOF. (ii) $\Rightarrow$ (i) follows from Lemma 3.

(i) $\Rightarrow$ (ii). Assume that G is a rigid grammar and $D \subseteq \mathrm{FL}(G)$. Then by Lemma 3, for some substitution σ, $\sigma[\mathrm{GF}(D)] \subseteq G$. $\sigma[\mathrm{GF}(D)]$ is a rigid grammar, so σ unifies all the types assigned to the same symbol by $\mathrm{GF}(D)$. Hence $\mathrm{RG}(D)$ exists and $\mathrm{RG}(D) = \sigma_0[\mathrm{GF}(D)]$, where σ_0 is a most general unifier that unifies all the types assigned to the same symbol by $\mathrm{GF}(D)$. This means that for some substitution τ, $\sigma = \tau \circ \sigma_0$. So $G \supseteq \sigma[\mathrm{GF}(D)] = (\tau \circ \sigma_0)[\mathrm{GF}(D)] = \tau[\sigma_0[\mathrm{GF}(D)]] = \tau[\mathrm{RG}(D)]$. ■

Corollary 5 *Let D be a finite set of F-structures. Then, for any rigid grammar G, if $D \subseteq \mathrm{FL}(G)$, then $\mathrm{FL}(\mathrm{RG}(D)) \subseteq \mathrm{FL}(G)$.*

PROOF. From Proposition 4, using Lemma 1. ■

Thus, the F-language of $\mathrm{RG}(D)$ is the smallest among the F-languages of rigid grammars which include D.

Corollary 6 *Let D and D' be finite sets of F-structures such that $D \subseteq D'$. Then, if $\mathrm{RG}(D')$ exists, $\mathrm{RG}(D)$ exists and $\tau[\mathrm{RG}(D)] \subseteq \mathrm{RG}(D')$ for some substitution τ and $\mathrm{FL}(\mathrm{RG}(D)) \subseteq \mathrm{FL}(\mathrm{RG}(D'))$.*

3.3.2 Adequacy of the Output

The output of RG has the property we call *adequacy*, and this fact will be crucial in proving the main theorem of this paper. In showing this, we prove some general facts about adequate rigid grammars as well.

Definition A rigid grammar G is called *minimal* if the following condition holds: for any rigid grammar G', if $\mathrm{FL}(G') = \mathrm{FL}(G)$ and $\sigma[G'] \subseteq G$ for some σ, then $\tau[G] \subseteq G'$ for some τ.

Note that, for rigid grammars G and G', if $\tau[G] \subseteq G'$ and $\sigma[G'] \subseteq G$, then $\tau[G] = G'$ and $\sigma[G'] = G$, that is, G and G' are alphabetic variants.

Lemma 7 *For every rigid grammar G, there is a minimal rigid grammar G' such that $\mathrm{FL}(G) = \mathrm{FL}(G')$ and $\sigma[G'] \subseteq G$ for some substitution σ.*

PROOF. Let G be any rigid grammar. Up to alphabetic variants, there are only a finite number of rigid grammars G' such that $\mathrm{FL}(G) = \mathrm{FL}(G')$ and $\sigma[G'] \subseteq G$ for some substitution σ. One of them must be minimal. ■

Minimality of a rigid grammar turns out to be equivalent to adequacy, which we now define.

Let \$ be a symbol not in Σ.

Definition For any grammar G, and for each $A \in \mathrm{SubTp}(G)$, let $G_A = G \cup \{\langle \$, A \rangle\}$. G_A is a grammar over the extended alphabet $\Sigma \cup \{\$\}$.

Definition Let G be any grammar. For each $A \in \mathrm{SubTp}(G)$, define two sets $\mathcal{U}_G(A)$ and $\mathcal{V}_G(A)$ of F-structures as follows. The latter is a set of F-structures over the extended alphabet $\Sigma \cup \{\$\}$.

- $\mathcal{U}_G(A) = \{\, T \mid G{:}T \mapsto A \,\}$,
- $\mathcal{V}_G(A) = \{\, T \mid T \in \mathrm{FL}(G_A)$ and \$ occurs exactly once in $T \,\}$.

Members of $\mathcal{V}_G(A)$ are 'contexts' in which an F-structure of type A can occur. If $T_1(\$) \in \mathcal{V}_G(A)$ and $T_2 \in \mathcal{U}_G(A)$, then $T_1(T_2) \in \mathrm{FL}(G)$, where $T_1(T_2)$ is the result of replacing the leaf labeled by \$ in $T_1(\$)$ by T_2.

Lemma 8 *The following hold:*

1. *If $\mathcal{U}_G(A) \neq \emptyset$ and either $\mathcal{U}_G(B/A) \neq \emptyset$ or $\mathcal{U}_G(A\backslash B) \neq \emptyset$, then $\mathcal{U}_G(B) \neq \emptyset$.*
2. *If $\mathcal{V}_G(B) \neq \emptyset$ and either $\mathcal{U}_G(B/A) \neq \emptyset$ or $\mathcal{U}_G(A\backslash B) \neq \emptyset$, then $\mathcal{V}_G(A) \neq \emptyset$.*
3. *If $\mathcal{V}_G(B) \neq \emptyset$ and $\mathcal{U}_G(A) \neq \emptyset$, then $\mathcal{V}_G(B/A) \neq \emptyset$ and $\mathcal{V}_G(A\backslash B) \neq \emptyset$.*

Definition A grammar G is called *adequate* if for each $A \in \mathrm{SubTp}(G)$, $\mathcal{U}_G(A) \neq \emptyset$ and $\mathcal{V}_G(A) \neq \emptyset$.

Our definition of 'adequate' differs from that of Buszkowski (1988), but the two coincide with respect to rigid grammars. An adequate categorial grammar is similar to a context-free grammar with no useless symbols.

Lemma 9 *A grammar G is adequate if and only if G satisfies the following two conditions:*

(i) *for any $A, B \in \mathrm{SubTp}(G)$, if $\mathcal{U}_G(B/A) \neq \emptyset$ or $\mathcal{U}_G(A\backslash B) \neq \emptyset$, then $\mathcal{U}_G(A) \neq \emptyset$.*

(ii) *for any $A \in \mathrm{SubTp}(G)$, $\mathcal{U}_G(A) \neq \emptyset$ implies $\mathcal{V}_G(A) \neq \emptyset$.*

Definition Note that any type A can be written uniquely in the following form:

$$(\ldots(p|A_1)|\ldots)|A_n$$

where $B|C$ stands for either B/C or $C\backslash B$, and $p \in \mathrm{Pr}$. For $0 \leq i \leq n$, we call the subtype $(\ldots(p|A_1)|\ldots)|A_i$ of A (when $i = 0$, we take this to be p) a *head subtype* of A. p is the *head* of A and is denoted $\mathrm{head}(A)$. A_i's are called *argument subtypes* of A.

Lemma 10 *If $\mathcal{U}_G(B) \neq \emptyset$, B occurs as a head subtype of some type in* $\mathrm{range}(G)$.

Proposition 11 *For every rigid grammar G, if G is minimal, G is adequate.*

PROOF. We prove the contrapositive. Suppose that G is a rigid grammar that is not adequate. Then there must be a type $A = (\ldots(p|A_1)|\ldots)|A_n \in \mathrm{range}(G)$ such that either $\mathcal{U}_G(A_i) = \emptyset$ for some i $(1 \leq i \leq n)$ or $\mathcal{V}_G(p) = \emptyset$. To see this, use Lemmas 8, 9, and 10.

CASE 1. $\mathcal{V}_G(p) = \emptyset$.

CASE 1A. $n = 0$ $(p = A)$. Let $G' = G - \{\langle a, A\rangle\}$, where a is a symbol such that $G{:}\, a \mapsto A$. It is clear that $\mathrm{FL}(G') = \mathrm{FL}(G)$, so G is not minimal.

CASE 1B. $n \geq 1$. Pick a fresh variable x and let σ be the substitution that maps x to $p|A_1$ (and every other variable to itself). For any type $B \in \mathrm{SubTp}(G)$, let $g(B)$ be the result of replacing all occurrences of $p|A_1$ in B by x. ($\sigma(g(B)) = B$.) Take the rigid grammar $G' = \{ \langle b, g(B) \rangle \mid \langle b, B \rangle \in G \}$. $G = \sigma[G']$, and there is no substitution τ such that $G' = \tau[G]$. $\mathrm{FL}(G') \subseteq \mathrm{FL}(G)$, so it remains to show $\mathrm{FL}(G) \subseteq \mathrm{FL}(G')$. Suppose $T \in \mathrm{FL}(G)$. Let h be a parse of T by G. Let h' be the function from the nodes of T to types such that $h'(\nu) = g(h(\nu))$. We show that h' is a parse of T by G'. That h' maps the root node of T to t is obvious. If ν is a leaf node of T labeled by the symbol b, $h'(b)$ is a type assigned to b by G', by the construction of G'. Let ν be any node of T with daughters ν_1 and ν_2. Then for some $B, C \in \mathrm{SubTp}(G)$, either (i) $h(\nu) = B$, $h(\nu_1) = B/C$, and $h(\nu_2) = C$, or (ii) $h(\nu) = B$, $h(\nu_1) = C$, and $h(\nu_2) = C\backslash B$. In the case of (i), since $\mathcal{V}_G(p) = \emptyset$, $B \neq p$, so $B/C \neq p|A_1$. Thus $h'(\nu) = g(B)$, $h'(\nu_1) = g(B/C) = g(B)/g(C)$, and $h'(C) = g(C)$. Case (ii) is similar. This shows that h' is a parse of T by G'. So $T \in \mathrm{FL}(G')$.

CASE 2. $\mathcal{U}_G(A_i) = \emptyset$. Pick a fresh variable x and let σ be the substitution that maps x to $(\ldots(p|A_1)|\ldots)|A_i$. For any type $B \in \mathrm{SubTp}(G)$, let $g(B)$ be the result of replacing all occurrences of $(\ldots(p|A_1)|\ldots)|A_i$ in B by x. ($\sigma(g(B)) = B$.) Take the rigid grammar $G' = \{ \langle b, g(B) \rangle \mid \langle b, B \rangle \in G \}$. That G' is a counterexample to the minimality of G can be shown in a similar way to Case 1b. ■

The converse of Proposition 11 also holds (Corollary 19).

Since the questions '$\mathcal{U}_G(A) \neq \emptyset$?' and '$\mathcal{V}_G(A) \neq \emptyset$?' are decidable, the proof of Proposition 11 provides an effective procedure which, given a rigid grammar G, finds an adequate (and minimal, by Corollary 19) rigid grammar G' such that $\mathrm{FL}(G) = \mathrm{FL}(G')$.

Corollary 12 *Given a rigid grammar G, one can effectively find an adequate rigid grammar G' such that $\mathrm{FL}(G) = \mathrm{FL}(G')$ and $\sigma[G'] \subseteq G$ for some substitution σ.*

Corollary 13 $\mathrm{RG}(D)$, *if it exists, is adequate.*

PROOF. By Proposition 4, $\mathrm{RG}(D)$, if it exists, is minimal, and hence adequate by Proposition 11. ■

3.3.3 Further Properties of RG

This section will not be needed in the proof of the main theorem and therefore can be skipped, but may be of interest in its own right.

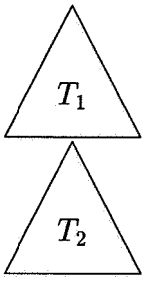
Is it the case that, for any rigid grammar G, there is some D such that $\mathrm{FL}(G) = \mathrm{FL}(\mathrm{RG}(D))$? The answer is positive. The following construction, essentially due to Sakakibara (1992), gives more information.

Definition Let G be an adequate grammar. For each $A \in \mathrm{SubTp}(G)$, pick two F-structures, $U_G(A)$ and $V_G(A)$ as follows:

- $U_G(A)$ is an F-structure of the minimal size in $\mathcal{U}_G(A)$.
- $V_G(A)$ is an F-structure of the minimal size in $\mathcal{V}_G(A)$.

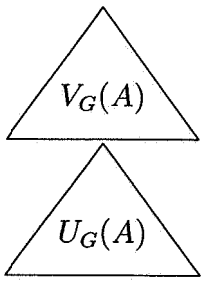
It is actually immaterial which members of $\mathcal{U}_G(A)$ and $\mathcal{V}_G(A)$ $U_G(A)$ and $V_G(A)$ are, except that $V_G(t)$ must be \$.

If T_1 is an F-structure over $\Sigma \cup \{\$\}$ with exactly one occurrence of \$, and T_2 is an F-structure over Σ, then we write

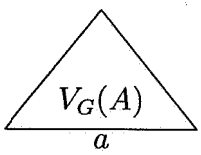


for the result of substituting T_2 for \$ in T_1.

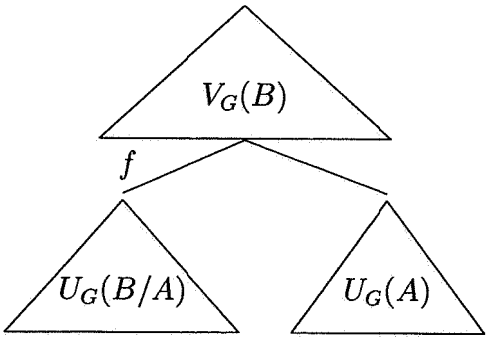
Definition Let G be an adequate rigid grammar. CS(G) (the *characteristic sample* of G) is the set which contains the following F-structures:

- 

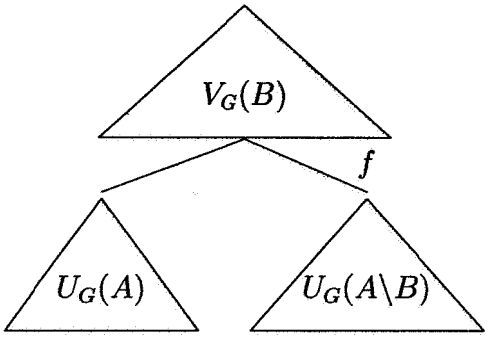
for all $A \in \mathrm{SubTp}(G)$,

- 

for all $a \in \Sigma$, $A \in \mathrm{range}(G)$ such that $G\colon a \mapsto A$.

- 

for all $B/A \in \mathrm{SubTp}(G)$,

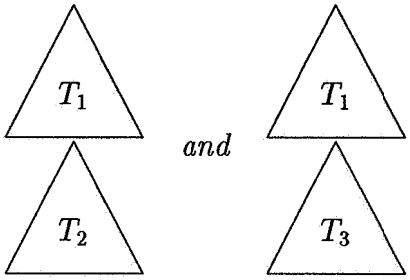
- 

for all $A\backslash B \in \mathrm{SubTp}(G)$.

Note $\mathrm{CS}(G) \subseteq \mathrm{FL}(G)$.

Lemma 14 *If G is a rigid grammar, G assigns at most one type to any F-structure, and each F-structure in $\mathrm{FL}(G)$ has a unique parse.*

Lemma 15 *Let G be a rigid grammar. Let T_1 be an F-structure over $\Sigma \cup \{\$\}$ with exactly one occurrence of \$, and let T_2 and T_3 be F-structures over Σ. If*



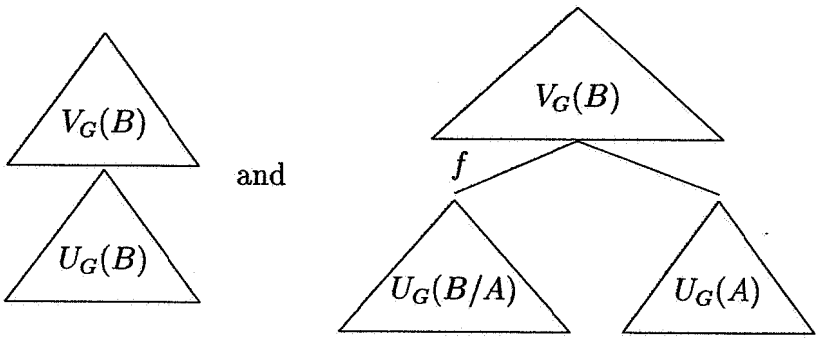
are both in $\mathrm{FL}(G)$, then G assigns the same type to T_2 and T_3.

PROOF. By induction on T_1. ∎

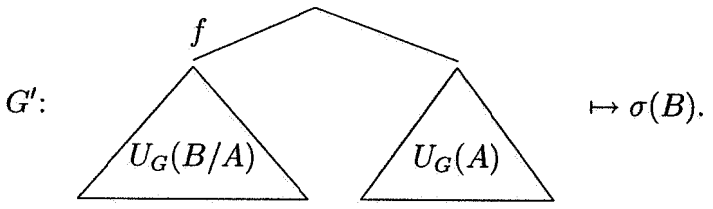
Proposition 16 *Let G be an adequate rigid grammar. For any rigid grammar G', if $\mathrm{CS}(G) \subseteq \mathrm{FL}(G')$, then $\sigma[G] \subseteq G'$ for some substitution σ.*

PROOF. Since $\mathrm{CS}(G) \subseteq \mathrm{FL}(G')$, G' assigns a type to each substructure of F-structures in $\mathrm{CS}(G)$. Let σ be the substitution that maps each variable x in $\mathrm{SubTp}(G)$ to the type G' assigns to $U_G(x)$.

We show by induction on $C \in \mathrm{SubTp}(G)$ that $G'{:}U_G(C) \mapsto \sigma(C)$. By the definition of σ, if x is a variable in $\mathrm{SubTp}(G)$, then $G'{:}U_G(x) \mapsto \sigma(x)$. Since $V_G(t) = \$$, $U_G(t)$ is in $\mathrm{CS}(G)$, and hence in $\mathrm{FL}(G')$, so $G'{:}U_G(t) \mapsto t$. Suppose $C = B/A \in \mathrm{SubTp}(G)$. By induction hypothesis, $G'{:}U_G(A) \mapsto \sigma(A)$ and $G'{:}U_G(B) \mapsto \sigma(B)$. Since

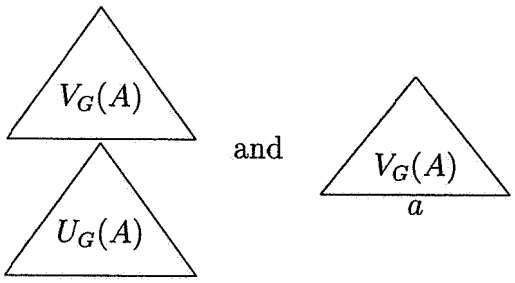


are both in $\mathrm{CS}(G)$, and hence in $\mathrm{FL}(G')$, by Lemma 15,



Then it must be that $G'\colon U_G(B/A) \mapsto \sigma(B)/\sigma(A)$. The case $C = A\backslash B$ is treated similarly.

Suppose $G\colon a \mapsto A$. Since $G'\colon U_G(A) \mapsto \sigma(A)$, and



are both in $\mathrm{CS}(G)$, and hence in $\mathrm{FL}(G')$, it must be that $G'\colon a \mapsto \sigma(A)$. Therefore, $\sigma[G] \subseteq G'$. ■

The following are immediate consequences of the above proposition.

Corollary 17 *Let G be an adequate rigid grammar. For any rigid grammar G', if $\mathrm{FL}(G) = \mathrm{FL}(G')$, then $\sigma[G] \subseteq G'$ for some substitution σ.*

Corollary 18 *For any rigid grammar G, there is, up to alphabetic variants, a unique adequate rigid grammar G' such that $\mathrm{FL}(G) = \mathrm{FL}(G')$.*

Corollary 19 *For every rigid grammar G, if G is adequate, G is minimal.*

Thus, we can use 'minimal' and 'adequate' interchangeably when talking about rigid grammars.

Corollary 20 *Let G be an adequate rigid grammar. For any finite set D of F-structures, if $\mathrm{CS}(G) \subseteq D \subseteq \mathrm{FL}(G)$, then $\mathrm{RG}(D)$ is an alphabetic variant of G.*

PROOF. From Propositions 4 and 16. ■

Corollary 21 *Given any rigid grammar G, a finite set D of F-structures such that $\mathrm{FL}(\mathrm{RG}(D)) = \mathrm{FL}(G)$ can be effectively found.*

PROOF. Given a rigid grammar G, an adequate rigid grammar G' such that $\mathrm{FL}(G) = \mathrm{FL}(G')$ can be effectively found (Corollary 12), and from G', the set $\mathrm{CS}(G')$ can be effectively constructed. Then the corollary follows from Corollary 20. ■

Definition Let φ_{RG} be the function from finite sequences of F-structures to rigid grammars such that $\varphi_{\mathrm{RG}}(\langle T_1, \ldots, T_n \rangle) = \mathrm{RG}(\{T_1, \ldots, T_n\})$.

If we take $\langle \mathcal{C}, \mathrm{FL}(\cdot) \rangle$ to be the grammar system, and if we equate grammars that are alphabetic variants, then Corollary 20 has the following consequence:

Theorem 3 *With respect to the grammar system $\langle \mathcal{C}, \mathrm{FL}(\cdot) \rangle$, φ_{RG} identifies $\mathcal{C}_1$ in the limit from positive data.*

PROOF. Let G be a rigid grammar and let $T_1, T_2, T_3, \ldots$ be an infinite sequence of F-structures such that $\{T_1, T_2, T_3, \ldots\} = \mathrm{FL}(G)$. There is an adequate rigid grammar G' such that $\mathrm{FL}(G) = \mathrm{FL}(G')$. There is an n such that for all $m \geq n$ $\mathrm{CS}(G') \subseteq \{T_1, \ldots, T_m\}$. Then by Corollary 20, for all $m \geq n$, $\varphi_{\mathrm{RG}}(\langle T_1, \ldots, T_m \rangle)$ is an alphabetic variant of G'. Thus, if we equate grammars that are alphabetic variants (or adjust φ_{RG} so that it outputs a certain representative), then φ_{RG} converges to an adequate rigid grammar G' such that $\mathrm{FL}(G) = \mathrm{FL}(G')$. ■

Theorem 3 is parallel to the result obtained by Sakakibara (1992) about context-free grammars: the class of what he calls 'reversible context-free grammars', viewed as tree-generating devices, is identifiable in the limit from positive data (consisting of trees). In fact, there is a close connection between reversible context-free grammars and rigid classical categorial grammars (an analogue of Lemma 15 holds for reversible context-free grammars), and the similarity of Sakakibara's algorithm and Buszkowski's is evident. The nice thing about reversible context-free grammars is that they provide a kind of normal form for context-free grammars in that every context-free language is generated by some reversible context-free grammar. On the other hand, this makes it impossible to identify the class of reversible context-free grammars, as generators of strings, in the limit from positive data, by Gold's theorem. In section 4, we show that RG can be used in an algorithm that identifies $\mathcal{C}_k$ in the limit from positive data, for each k.

3.4 An Incremental Version of RG

When applying RG successively to a sequence of increasing data $D_1 \subset D_2 \subset D_3 \subset \cdots$, it is more efficient to make use of the previous value $\mathrm{RG}(D_{i-1})$ to compute the current value $\mathrm{RG}(D_i)$.

Definition If G is a grammar and D is a finite set of F-structures, $\mathrm{RG}_2(G, D)$ is defined to be the rigid grammar $\sigma[G' \cup \mathrm{GF}(D)]$, where G' is an alphabetic variant of G such that G' and $\mathrm{GF}(D)$ do not employ common variables, and σ is a most general unifier of the class $\{\{A \mid G' : a \mapsto A\} \cup \{A \mid \mathrm{GF}(D) : a \mapsto A\} \mid a \in \Sigma\}$, if such exists.

Lemma 22 *Let G be any grammar and D be a finite set of F-structures. For any rigid grammar G', the following are equivalent:*

(i) $\sigma[G] \subseteq G'$ *for some substitution* σ *and* $D \subseteq \mathrm{FL}(G')$.

(ii) $\mathrm{RG}_2(G, D)$ *exists and* $\tau[\mathrm{RG}_2(G, D)] \subseteq G'$ *for some substitution* τ.

PROOF. (ii) $\Rightarrow$ (i) is easy, using Lemma 3.

(i) $\Rightarrow$ (ii). By Lemma 3, $\sigma_0[\mathrm{GF}(D)] \subseteq G'$ for some σ_0. Assuming that G does not involve any variables used in $\mathrm{GF}(D)$ (if it does, use an alphabetic variant of G), we have $G' \supseteq \sigma[G] \cup \sigma_0[\mathrm{GF}(D)] = \sigma_1[G \cup \mathrm{GF}(D)]$, where σ_1 agrees with σ on variables that appear in G and with σ_0 on variables that appear in $\mathrm{GF}(D)$. Since G' is rigid, this means that σ_1 unifies the types assigned to the same symbol by $G \cup \mathrm{GF}(D)$. So $\mathrm{RG}_2(G, \mathrm{GF}(D))$ exists, and if it is $\sigma_2[G \cup \mathrm{GF}(D)]$, $\sigma_1 = \tau \circ \sigma_2$ for some substitution τ. Therefore, $G' \supseteq \tau[\mathrm{RG}_2(G, \mathrm{GF}(D))]$. ∎

Lemma 23 $\mathrm{RG}_2(\mathrm{RG}(D_1), D_2) = \mathrm{RG}(D_1 \cup D_2)$, *up to alphabetic variants.*

PROOF. Suppose that $\mathrm{RG}_2(\mathrm{RG}(D_1), D_2)$ exists. By Lemma 22, $\sigma_1[\mathrm{RG}(D_1)] \subseteq \mathrm{RG}_2(\mathrm{RG}(D_1), D_2)$ for some substitution σ_1 and $D_2 \subseteq \mathrm{FL}(\mathrm{RG}_2(\mathrm{RG}(D_1), D_2))$. This means $D_1 \cup D_2 \subseteq \mathrm{FL}(\mathrm{RG}_2(\mathrm{RG}(D_1), D_2))$, and so by Proposition 4, $\mathrm{RG}(D_1 \cup D_2)$ exists, and $\tau_1[\mathrm{RG}(D_1 \cup D_2)] \subseteq \mathrm{RG}_2(\mathrm{RG}(D_1), D_2)$ for some τ_1.

Suppose now that $\mathrm{RG}(D_1 \cup D_2)$ exists. By Corollary 6, $\mathrm{RG}(D_1)$ exists and $\sigma_2[\mathrm{RG}(D_1)] \subseteq \mathrm{RG}(D_1 \cup D_2)$ for some substitution σ_2. Since $D_2 \subseteq \mathrm{FL}(\mathrm{RG}(D_1 \cup D_2))$, this means, by Lemma 22, that $\mathrm{RG}_2(\mathrm{RG}(D_1), D_2)$ exists and for some substitution τ_2, $\tau_2[\mathrm{RG}_2(\mathrm{RG}(D_1), D_2)] \subseteq \mathrm{RG}(D_1 \cup D_2)$.

We have proved that if one of $\mathrm{RG}_2(\mathrm{RG}(D_1), D_2)$ and $\mathrm{RG}(D_1 \cup D_2)$ exists the other exists and they are alphabetic variants. ∎

RG_2 will be used in the proof of the main theorem.

4 Proof of the Main Theorem

4.1 The Case $k = 1$

We first prove the main theorem for the special case $k = 1$, to illustrate the core idea of the proof.

We define a computable function $\varphi_{\mathcal{C}_1}$ that identifies $\mathcal{C}_1$ in the limit as the composition of two computable functions, $\psi_{\mathcal{C}_1}$ and χ. $\psi_{\mathcal{C}_1}$ is a function from finite sequences of strings to finite sets of rigid grammars, and χ is a function that takes two arguments, a finite set of rigid grammars and a positive integer, and returns a member of the first argument.

When applied successively to larger and larger initial segments of an enumeration of the language of some rigid grammar, $\psi_{\mathcal{C}_1}$ converges to a finite set which contains a correct grammar for the language. Given a finite set of grammars, what χ does is to 'compute in the limit' a grammar in that set which generates the smallest language.

Definition $\psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_n \rangle)$ is defined recursively as follows. Let $\mathcal{G}_n$ abbreviate $\psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_n \rangle)$.

$$\begin{aligned} \mathcal{G}_0 &= \{\emptyset\} \\ \mathcal{G}_{n+1} &= \{\, G \in \mathcal{G}_n \mid s_{n+1} \in \mathrm{L}(G) \,\} \cup \\ &\quad \{\, G' \mid \text{for some } G \in \mathcal{G}_n \text{ and some F-structure } T, \\ &\qquad s_{n+1} \notin \mathrm{L}(G),\ s_{n+1} = \mathrm{yield}(T), \text{ and } G' = \mathrm{RG}_2(G, \{T\}) \,\} \end{aligned}$$

Lemma 24 *If $\mathcal{G} \in \mathrm{range}(\psi_{\mathcal{C}_1})$, then $\mathcal{G}$ is a finite set of adequate rigid grammars.*

PROOF. That any $\mathcal{G} \in \mathrm{range}(\psi_{\mathcal{C}_1})$ is finite is obvious, since for any string s, there are only finitely many F-structures T such that $\mathrm{yield}(T) = s$. By induction on n, one can show that, if $G \in \psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_n \rangle)$, G is an alphabetic variant of $\mathrm{RG}(D)$ for some D, using Lemma 23. Then by Corollary 13, G is adequate. ∎

Lemma 25 *If $G \in \psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_n \rangle)$, then $\{s_1, \ldots, s_n\} \subseteq \mathrm{L}(G)$.*

PROOF. By induction on n. ∎

Lemma 26 *If G' is a rigid grammar such that $\{s_1, \ldots, s_n\} \subseteq \mathrm{L}(G')$, then there is a $G \in \psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_n \rangle)$ such that $\sigma[G] \subseteq G'$ for some σ.*

PROOF. Induction on n. Let G' be a rigid grammar. The case $n = 0$ is trivial. Suppose $\{s_1, \ldots, s_{n+1}\} \subseteq \mathrm{L}(G')$. By induction hypothesis, there is a $G \in \psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_n \rangle)$ such that $\sigma[G] \subseteq G'$ for some σ. If $s_{n+1} \in \mathrm{L}(G)$, then $G \in \psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_{n+1} \rangle)$. If $s_{n+1} \notin G$, let T be an F-structure such that $\mathrm{yield}(T) = s_{n+1}$ and $T \in \mathrm{FL}(G')$. Then by Lemma 22, $\mathrm{RG}_2(G, \{T\})$ exists and $\mathrm{RG}_2(G, \{T\}) \in \psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_{n+1} \rangle)$, and there is a substitution τ such that $\tau[\mathrm{RG}_2(G, \{T\})] \subseteq G'$. ∎

Lemma 27 *Let $s_1, s_2, s_3, \ldots$ be an infinite sequence of strings. If G' is a rigid grammar such that $\{s_1, s_2, s_3, \ldots\} = \mathrm{L}(G')$, then for some n, there is a rigid grammar $G \in \psi_{C_1}(\langle s_1, \ldots, s_n \rangle)$ such that $\mathrm{L}(G) = \mathrm{L}(G')$ and $\sigma[G] \subseteq G'$ for some substitution σ.*

PROOF. Suppose that G' is a rigid grammar such that $\{s_1, s_2, s_3, \ldots\} = \mathrm{L}(G')$. By Lemma 26, for every n, there is a $G \in \psi_{C_1}(\langle s_1, \ldots, s_n \rangle)$ such that $\sigma[G] \subseteq G'$ for some σ. $\sigma[G] \subseteq G'$ implies $\mathrm{L}(G) \subseteq \mathrm{L}(G')$. If $\mathrm{L}(G) \subset \mathrm{L}(G')$, then, by Lemma 25, there is an n such that for all $m \geq n$, $G \notin \psi_{C_1}(\langle s_1, \ldots, s_m \rangle)$. Since up to alphabetic variants there are only finitely many rigid grammars G such that $\sigma[G] \subseteq G'$ for some σ, it follows that there is an n such that for all G, if $\sigma[G] \subseteq G'$ for some σ and $\mathrm{L}(G) \subset \mathrm{L}(G')$, $G \notin \psi_{C_1}(\langle s_1, \ldots, s_n \rangle)$. Then the required conclusion follows. ■

Lemma 28 (Convergence Lemma) *Let $s_1, s_2, s_3, \ldots$ be an infinite sequence of strings. Then there are some n and a finite set $\mathcal{G}$ of rigid grammars such that for all $m \geq n$, $\psi_{C_1}(\langle s_1, \ldots, s_m \rangle) = \mathcal{G}$.*

PROOF. Corresponding to the definition of $\mathcal{G}_n = \psi_{C_1}(\langle s_1, \ldots, s_n \rangle)$, one can form trees $\mathcal{T}_n$ in the following way, each of whose nodes is labeled by either a grammar in $\mathcal{G}_0 \cup \cdots \cup \mathcal{G}_n$ or a special symbol ∂.

Stage 0. $\mathcal{T}_0$ is a tree with just one node, which is labeled by the empty grammar $\emptyset$.

Stage $n+1$. $\mathcal{T}_{n+1}$ is obtained by attaching daughter nodes to some leaf nodes (possibly none) of $\mathcal{T}_n$ in the following way. For each leaf node ν of $\mathcal{T}_n$, do the following:

1. If ν is labeled by ∂, do nothing and leave it as a leaf node.
2. If ν is labeled by a grammar G such that $s_{n+1} \in \mathrm{L}(G)$, do nothing and leave it as a leaf node.
3. If ν is labeled by a grammar G such that $s_{n+1} \notin \mathrm{L}(G)$, let

   $$\mathcal{F} = \{\, G' \mid G' = \mathrm{RG}_2(G, \{T\}) \text{ for some } T \text{ such that } \mathrm{yield}(T) = s_{n+1} \,\}$$

   and

   (a) if $\mathcal{F} = \emptyset$, create a new node ν', label it by ∂, and attach it to ν as its only daughter.

   (b) if $\mathcal{F} = \{G_1, \ldots, G_l\} \neq \emptyset$, then create a new node ν_i for each $G_i \in \mathcal{F}$, label it by G_i, and attach $\nu_1, \ldots, \nu_l$ to ν as its daughters.

It is easy to see the following:

(I) $\mathcal{G}_n = \{\, G \mid G \text{ labels a leaf node of } \mathcal{T}_n \,\}$.

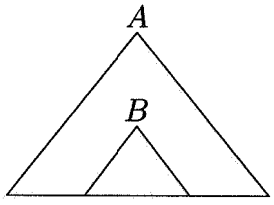
(II) If in $\mathcal{T}_n$ G' labels a daughter ν' of a node ν labeled by G, $\sigma[G] \subseteq G'$ for some σ, but G and G' are not alphabetic variants.

Since each $\mathcal{T}_{n+1}$ is an 'extension' of $\mathcal{T}_n$, one can construct the union $\mathcal{T}$ of all $\mathcal{T}_n$'s. Indeed, the above definition of $\mathcal{T}_n$ describes an infinite procedure to construct $\mathcal{T}$. That is, the nodes of $\mathcal{T}$ are nodes that are created at some stage of the above procedure, and a node ν of $\mathcal{T}$ has ν' as one of its daughters if and only if at some stage n of the above procedure ν' is newly created and attached to ν as a daughter. Since for each node there is at most one stage at which daughters are attached to it, $\mathcal{T}$ is a finitely branching tree.

Claim $\mathcal{T}$ has no infinite branch.

This implies that $\mathcal{T}$ is finite by König's Lemma. To prove the claim, we need some definitions and a technical lemma on adequate rigid grammars.

Definition Let G be a rigid grammar and let $A, B \in \mathrm{SubTp}(G)$. We say that A *depends on* B (in G) if for all $T \in \mathcal{U}_G(A)$, there is a proper substructure T' of T ($T' \neq T$) such that $T' \in \mathcal{U}_G(B)$.



Definition Define the *degree* $d(G)$ of a grammar G as follows:

$$d(G) = |\mathrm{Var} \cap \mathrm{SubTp}(G)| + |\Sigma - \mathrm{dom}(G)|.$$

Definition Let $\mathrm{Head}(G)$ denote the set $\{\, \mathrm{head}(A) \mid A \in \mathrm{range}(G) \,\}$ for any G.

Lemma 29 (Key Lemma) *Let G and G' be adequate rigid grammars. If $\sigma[G] \subseteq G'$ and G and G' are not alphabetic variants, then $d(G) > d(G')$.*

PROOF. Suppose that G and G' are adequate rigid grammars such that $\sigma[G] \subseteq G'$. Then, using Lemma 10,

$$\begin{aligned}
|\mathrm{Var} \cap \mathrm{SubTp}(G')| &= |\mathrm{Head}(G') - \{t\}| \\
&= |(\mathrm{Head}(\sigma[G]) - \{t\}) \cup (\mathrm{Head}(G' - \sigma[G]) - \{t\})| \\
&\leq |\mathrm{Head}(\sigma[G]) - \{t\}| + |\mathrm{Head}(G' - \sigma[G]) - \{t\}| \quad (1) \\
&\leq |\mathrm{Head}(\sigma[G]) - \{t\}| + |\mathrm{Head}(G' - \sigma[G])| \quad (2) \\
&\leq |\mathrm{Head}(\sigma[G]) - \{t\}| + |\mathrm{dom}(G') - \mathrm{dom}(G)| \quad (3) \\
&\leq |\mathrm{Head}(G) - \{t\}| + |\mathrm{dom}(G') - \mathrm{dom}(G)| \quad (4) \\
&= |\mathrm{Var} \cap \mathrm{SubTp}(G)| + |\mathrm{dom}(G') - \mathrm{dom}(G)|
\end{aligned}$$

where equality holds for

(1) just in case $(\mathrm{Head}(\sigma[G]) - \{t\}) \cap (\mathrm{Head}(G' - \sigma[G]) - \{t\}) = \emptyset$.

(2) just in case $t \notin \mathrm{Head}(G' - \sigma[G])$.

(3) just in case for all $b, c \in \mathrm{dom}(G') - \mathrm{dom}(G)$, $b \neq c$ implies $\mathrm{head}(B) \neq \mathrm{head}(C)$, where B and C are the types such that $G' : b \mapsto B$ and $G' : c \mapsto C$.

(4) just in case for all $x \in \mathrm{Head}(G) - \{t\}$, $\mathrm{head}(\sigma(x)) \neq t$ and for all $x, y \in \mathrm{Head}(G) - \{t\}$, $x \neq y$ implies $\mathrm{head}(\sigma(x)) \neq \mathrm{head}(\sigma(y))$.

So $|\mathrm{Var} \cap \mathrm{SubTp}(G')| \leq |\mathrm{Var} \cap \mathrm{SubTp}(G)| + |\mathrm{dom}(G') - \mathrm{dom}(G)|$, which is equivalent to $d(G') \leq d(G)$.

Assume that $d(G) = d(G')$. Then the conditions (1)–(4) above must hold. Let

$$\begin{aligned} \{x_1, \ldots, x_m\} &= \mathrm{Head}(G) - \{t\}, \\ \{y_1, \ldots, y_l\} &= \mathrm{Head}(G' - \sigma[G]), \\ \mathrm{head}(\sigma(x_i)) &= z_i. \end{aligned}$$

$\{z_1, \ldots, z_m, y_1, \ldots, y_l\} = \mathrm{Head}(G') - \{t\}$ and $z_1, \ldots, z_m, y_1, \ldots, y_l$ are all distinct. We will show

(i) $\{y_1, \ldots, y_l\} = \emptyset$

(ii) $\sigma(x_i) = z_i$ for $1 \leq i \leq m$.

Note that $y_i \notin \{\sigma(x_1), \ldots, \sigma(x_m)\}$ and if $\sigma(x_i) \neq z_i$, then $z_i \notin \{\sigma(x_1), \ldots, \sigma(x_m)\}$. So suppose that there is some $w \in \{z_1, \ldots, z_m, y_1, \ldots, y_l\}$ such that $w \notin \{\sigma(x_1), \ldots, \sigma(x_m)\}$, to derive a contradiction. Since G' is adequate, $\mathcal{V}_{G'}(w) \neq \emptyset$, so w must occur as an argument subtype of some type A in $\mathrm{range}(G')$. There are two cases.

CASE 1. $A \in \mathrm{range}(G' - \sigma[G])$. Then A must look like

$$\ldots((\ldots(y_j|A_1)|\ldots)|w)|\ldots$$

Since y_j does not occur as the head of any $B \neq A$ in $\mathrm{range}(G')$, this implies that y_j depends on w.

CASE 2. $A \in \mathrm{range}(\sigma[G])$. A looks like

$$\ldots((\ldots(p|A_1)|\ldots)|w)|\ldots$$

$A = \sigma(B)$ for some $B \in \mathrm{range}(G)$, and the assumption $\sigma(x_i) \neq w$ for all x_i implies that $w \neq \sigma(C)$ for any $C \in \mathrm{SubTp}(G)$. This means that w occurs as an argument subtype of $\sigma(x_j)$, where $x_j = \mathrm{head}(B)$. Then $z_j = \mathrm{head}(\sigma(x_j)) \neq \sigma(x_j)$. z_j must depend on w, since every $D \in \mathrm{range}(G')$ with $\mathrm{head}(D) = z_j$ has $\sigma(x_j)$ as a head subtype.

Thus we have found a $w' \in \{z_1, \ldots, z_m, y_1, \ldots, y_l\}$ such that $w' \notin \{\sigma(x_1), \ldots, \sigma(x_m)\}$ and w' depends on w. Repeating this argument, we find

a cycle of dependency $w_0, w_1, \ldots, w_n$ $(n \geq 1)$ such that $w_0 = w_n$ and w_i depends on w_{i-1} for $1 \leq i \leq n$. This is a contradiction, for no adequate grammar can afford to have such a cycle of dependency; $\mathcal{U}_{G'}(w_0)$ would have to be empty.

So we have proved (i) and (ii). The conclusion that G and G' are alphabetic variants follows easily. ■

We now get back to the proof of the Convergence Lemma. $d(G) = |\Sigma|$ if $G = \emptyset$, and, by points (I), (II) above and Lemmas 24 and 29, this number must decrease as we go down in $\mathcal{T}$. So the length of each branch in $\mathcal{T}$ is bounded by $|\Sigma| + 1$, which proves the claim above.

Since $\mathcal{T}$ is the union of all $\mathcal{T}_n$'s and $\mathcal{T}_{n+1}$ is at least as large as $\mathcal{T}_n$, the finiteness of $\mathcal{T}$ implies that the sequence $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots$ 'stops growing' at some point. That is, there is some n such that for all $m \geq n$, $\mathcal{T}_m = \mathcal{T}_n$. Therefore $\mathcal{G}_n$ also converges. ■

Note that, in general, an adequate rigid grammar G with $d(G) \geq 1$ can have infinitely many non-equivalent adequate rigid grammars G' such that $\sigma[G] \subseteq G'$ for some σ. What the Key Lemma says is that there is no 'infinite chain' of non-equivalent adequate rigid grammars $G_0, G_1, G_2, \ldots$ such that for each n, $\sigma[G_n] \subseteq G_{n+1}$ for some σ.

Lemma 30 *Let G' be a rigid grammar and let $s_1, s_2, s_3, \ldots$ be an infinite sequence of strings such that $\{s_1, s_2, s_3, \ldots\} = \mathrm{L}(G')$. Then $\psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_n \rangle)$ converges to some finite set $\mathcal{G}$ of rigid grammars such that*

(i) *for some $G \in \mathcal{G}$, $\mathrm{L}(G') = \mathrm{L}(G)$,*

(ii) *for all $G \in \mathcal{G}$, $\mathrm{L}(G') \subseteq \mathrm{L}(G)$.*

PROOF. Part (ii) follows from Lemma 25. Part (i) follows from Lemma 27 and the fact that if $G \in \psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_n \rangle)$ and $s_{n+1} \in \mathrm{L}(G)$, then $G \in \psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_{n+1} \rangle)$. ■

If μ is a function that maps a finite set of rigid grammars $\mathcal{G}$ to one of its members G such that for all $G' \in \mathcal{G}$, $\mathrm{L}(G) \subseteq \mathrm{L}(G')$, then by Lemma 30, $\mu \circ \psi_{\mathcal{C}_1}$ identifies $\mathcal{C}_1$ in the limit. But we do not know that μ is computable, so this only shows that $\mathcal{C}_1$ is non-effectively identifiable in the limit. However, decidability of the membership problem '$s \in \mathrm{L}(G)$?' implies that the question '$\mathrm{L}(G) \subseteq \mathrm{L}(G')$?' is 'computable in the limit' and so is μ. This is enough to establish that $\mathcal{C}_1$ is identified in the limit by a computable function.

Definition Let $\Sigma^{\leq l}$ denote the set of strings over Σ with length $\leq l$.

Fix a particular recursive total ordering of all grammars. Below, 'first' means first in this ordering.

Definition Let $\mathcal{G}$ range over finite sets of grammars, and let n range over positive integers. Let χ be the computable function from finite sets of grammars and positive integers to grammars such that $\chi(\mathcal{G}, n)$ is the first grammar $G \in \mathcal{G}$ with the following property, if there is one:

$$\text{for all } G' \in \mathcal{G},\ \mathrm{L}(G) \cap \Sigma^{\leq n} \subseteq \mathrm{L}(G') \cap \Sigma^{\leq n}.$$

If there is no such G, let $\chi(\mathcal{G}, n)$ be undefined.

Lemma 31 *Let $\mathcal{G}$ be a finite set of grammars such that there is a $G \in \mathcal{G}$ such that for all $G' \in \mathcal{G}$, $\mathrm{L}(G) \subseteq \mathrm{L}(G')$. Let G_1 be the first such $G \in \mathcal{G}$. Then $\chi(\mathcal{G}, n)$ converges to G_1, that is, there is some n such that for all $m \geq n$, $\chi(\mathcal{G}, m) = G_1$.*

PROOF. For every $G' \in \mathcal{G}$ such that $\mathrm{L}(G_1) \subset \mathrm{L}(G')$, there is an $l_{G'}$ such that $\mathrm{L}(G_1) \cap \Sigma^{\leq l_{G'}} \subset \mathrm{L}(G') \cap \Sigma^{\leq l_{G'}}$. Let $n = \max(\{\, l_{G'} \mid G' \in \mathcal{G}, \mathrm{L}(G_1) \cap \Sigma^{\leq l_{G'}} \subset \mathrm{L}(G') \cap \Sigma^{\leq l_{G'}} \,\})$. Then it is clear that for all $m \geq n$, $\chi(\mathcal{G}, m) = G_1$. ∎

Definition Let $\varphi_{\mathcal{C}_1}(\langle s_1, \ldots, s_n \rangle) = \chi(\psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_n \rangle), n)$. ($n$ is the length of $\langle s_1, \ldots, s_n \rangle$.)

Theorem 2a *$\varphi_{\mathcal{C}_1}$ identifies $\mathcal{C}_1$ in the limit from positive data.*

PROOF. Let G be a rigid grammar and let $s_1, s_2, s_3, \ldots$ be an infinite sequence of strings such that $\{s_1, s_2, s_3, \ldots\} = \mathrm{L}(G)$. By Lemma 30, there is a number n_1, a finite set $\mathcal{G}$ of grammars, and a member G_1 of $\mathcal{G}$ such that for all $m \geq n_1$, $\psi_{\mathcal{C}_1}(\langle s_1, \ldots, s_m \rangle) = \mathcal{G}$, $\mathrm{L}(G_1) = \mathrm{L}(G)$, and G_1 is the first grammar in $\mathcal{G}$ such that for all $G' \in \mathcal{G}$, $\mathrm{L}(G_1) \subseteq \mathrm{L}(G')$. By Lemma 31, there is a number n_2 such that for all $m \geq n_2$, $\chi(\mathcal{G}, m) = G_1$. Let $n = \max(n_1, n_2)$. Then for all $m \geq n$, $\varphi_{\mathcal{C}_1}(\langle s_1, \ldots, s_m \rangle) = G_1$. ∎

4.2 The General Case

Having treated the special case $k = 1$, we now go on to prove our main theorem for the general case. Let k be a fixed positive integer. We will present an algorithm $\varphi_{\mathcal{C}_k}$ that identifies $\mathcal{C}_k$ in the limit from positive data. The idea is to associate with a k-valued grammar a rigid grammar over a 'disambiguated' alphabet.

Definition Let Υ be the alphabet that contains k copies of each symbol in Σ, that is, $\Upsilon = \bigcup\{\, \{a_1, \ldots, a_k\} \mid a \in \Sigma \,\}$.

Definition Let amb: $\Upsilon^+ \to \Sigma^+$ be the homomorphism that maps each copy a_i of a to a, for all $a \in \Sigma$. A string $u \in \Upsilon^+$ is called a *disambiguation* of a string $s \in \Sigma^+$ iff $s = \mathrm{amb}(u)$. $\mathrm{amb}(u)$ is called the *ambiguation* of u.

Definition Let $M \subseteq \Upsilon^+$. Then $\mathrm{amb}[M]$ denotes $\{\, \mathrm{amb}(u) \mid u \in M \,\}$.

Definition Let G be a k-valued grammar over Σ. A rigid grammar H over Υ is called a *disambiguation* of G if $G = \{ \langle \mathrm{amb}(b), B\rangle \mid \langle b, B\rangle \in H \}$. In such a case, G is called the *ambiguation* of H and is written $\mathrm{amb}[H]$.

Lemma 32 *For every k-valued grammar G over Σ, there is a rigid grammar H over Υ such that H is a disambiguation of G.*

Lemma 33 *If H is a rigid grammar over Υ, $\mathrm{amb}[\mathrm{L}(H)] = \mathrm{L}(\mathrm{amb}[H])$.*

Definition An F-structure U over Υ is called an *analysis* of a string $s \in \Sigma^+$, if $\mathrm{amb}(\mathrm{yield}(U)) = s$.

We can now define the guessing part ψ_{C_k} of the algorithm φ_{C_k}.

Definition ψ_{C_k} is a function that maps finite sequences of strings over Σ to finite sets of rigid grammars over Υ and is defined recursively as follows. Let $\mathcal{H}_n$ abbreviate $\psi_{C_k}(\langle s_1, \ldots, s_n\rangle)$.

$$\begin{aligned}
\mathcal{H}_0 &= \{\emptyset\} \\
\mathcal{H}_{n+1} &= \{ H \in \mathcal{H}_n \mid s_{n+1} \in \mathrm{amb}[\mathrm{L}(H)] \} \cup \\
&\quad \{ H' \in \mathcal{H}_n \mid \text{for some } H \in \mathcal{H}_n \text{ and some analysis } U \text{ of } s_{n+1}, \\
&\qquad\qquad s_{n+1} \notin \mathrm{amb}[\mathrm{L}(H)] \text{ and } H' = \mathrm{RG}_2(H, \{U\}) \}
\end{aligned}$$

The following two lemmas are analogous to Lemmas 24 and 25 and are proved easily.

Lemma 34 *If $\mathcal{H} \in \mathrm{range}(\psi_{C_k})$, then $\mathcal{H}$ is a finite set of adequate rigid grammars.*

Lemma 35 *If $H \in \psi_{C_k}(\langle s_1, \ldots, s_n\rangle)$, then $\{s_1, \ldots, s_n\} \subseteq \mathrm{amb}[\mathrm{L}(H)]$.*

Lemma 36 *Let G' be a k-valued grammar and let H' be a disambiguation of G'. If $\{s_1, \ldots, s_n\} \subseteq \mathrm{L}(G')$, there is an $H \in \psi_{C_k}(\langle s_1, \ldots, s_n\rangle)$ such that $\sigma[H] \subseteq H'$ for some substitution σ.*

PROOF. By induction on n. The case $n = 0$ is trivial. Suppose $\{s_1, \ldots, s_{n+1}\} \subseteq \mathrm{L}(G')$. By induction hypothesis, there is an $H \in \psi_{C_k}(\langle s_1, \ldots, s_n\rangle)$ such that $\sigma[H] \subseteq H'$ for some σ. If $s_{n+1} \in \mathrm{amb}[\mathrm{L}(H)]$, then $H \in \psi_{C_k}(\langle s_1, \ldots, s_n\rangle)$. Suppose $s_{n+1} \notin \mathrm{amb}[\mathrm{L}(H)]$. Since $s_{n+1} \in \mathrm{L}(G')$, by Lemma 33, there is a disambiguation u of s_{n+1} such that $u \in \mathrm{L}(H')$. Let U be an F-structure such that $\mathrm{yield}(U) = u$ and $U \in \mathrm{FL}(H')$. U is an analysis of s_{n+1}. Then by Lemma 22, $\mathrm{RG}_2(H, \{U\})$ exists and $\mathrm{RG}_2(H, \{U\}) \in \psi_{C_k}(\langle s_1, \ldots, s_{n+1}\rangle)$, and there is a substitution τ such that $\tau[\mathrm{RG}_2(H, \{U\})] \subseteq H'$. ■

Lemma 37 *Let G' be a k-valued grammar and let H' be a disambiguation of G'. Let $s_1, s_2, s_3, \ldots$ be an infinite sequence of strings over Σ such that $\{s_1, s_2, s_3, \ldots\} = \mathrm{L}(G')$. Then for some n, there is an $H \in \psi_{\mathcal{C}_k}(\langle s_1, \ldots, s_n \rangle)$ such that $\mathrm{amb}[\mathrm{L}(H)] = \mathrm{amb}[\mathrm{L}(H')]$ and $\sigma[H] \subseteq H'$ for some substitution σ.*

PROOF. By Lemma 36, for every n, there is an $H \in \psi_{\mathcal{C}_k}(\langle s_1, \ldots, s_n \rangle)$ such that $\sigma[H] \subseteq H'$ for some σ. $\sigma[H] \subseteq H'$ implies $\mathrm{amb}[\mathrm{L}(H)] \subseteq \mathrm{amb}[\mathrm{L}(H')]$. If $\mathrm{amb}[\mathrm{L}(H)] \subset \mathrm{amb}[\mathrm{L}(H')] = \mathrm{L}(G')$, then, by Lemma 35, there is an n such that for all $m \geq n$, $H \notin \psi_{\mathcal{C}_k}(\langle s_1, \ldots, s_m \rangle)$. Since up to alphabetic variants there are only finitely many rigid grammars H such that $\sigma[H] \subseteq H'$ for some σ, by taking n large enough, we can ensure $H \notin \psi_{\mathcal{C}_k}(\langle s_1, \ldots, s_n \rangle)$ for all H such that $\sigma[H] \subseteq H'$ and $\mathrm{amb}[\mathrm{L}(H)] \subset \mathrm{amb}[\mathrm{L}(H')]$, so the required conclusion follows. ∎

That $\psi_{\mathcal{C}_k}$ converges can be proved in an entirely parallel way to the case of $\psi_{\mathcal{C}_1}$.

Lemma 38 *Let $s_1, s_2, s_3, \ldots$ be an infinite sequence of strings over Σ. Then there are some n and a finite set $\mathcal{H}$ of rigid grammars over Υ such that for all $m \geq n$, $\psi_{\mathcal{C}_k}(\langle s_1, \ldots, s_m \rangle) = \mathcal{H}$.*

Lemma 39 *Let G' be a k-valued grammar over Σ and let $s_1, s_2, s_3, \ldots$ be an infinite sequence of strings such that $\{s_1, s_2, s_3, \ldots\} = \mathrm{L}(G')$. Then $\psi_{\mathcal{C}_k}(\langle s_1, \ldots, s_n \rangle)$ converges to some finite set $\mathcal{H}$ of rigid grammars over Υ such that*

(i) *for some $H \in \mathcal{H}$, $\mathrm{L}(G') = \mathrm{amb}[\mathrm{L}(H)]$,*

(ii) *for all $H \in \mathcal{H}$, $\mathrm{L}(G') \subseteq \mathrm{amb}[\mathrm{L}(H)]$.*

PROOF. From Lemmas 35 and 37 and the fact that $H \in \psi_{\mathcal{C}_k}(\langle s_1, \ldots, s_n \rangle)$ and $s_{n+1} \in \mathrm{amb}[\mathrm{L}(H)]$ imply $H \in \psi_{\mathcal{C}_k}(\langle s_1, \ldots, s_{n+1} \rangle)$. ∎

Definition Let $\mathcal{H}$ be a finite set of rigid grammars over Υ. Define $\mathrm{amb}[\mathcal{H}]$ to be $\{\, \mathrm{amb}[H] \mid H \in \mathcal{H} \,\}$.

Lemma 40 *Let G' be a k-valued grammar over Σ and let $s_1, s_2, s_3, \ldots$ be an infinite sequence of strings such that $\{s_1, s_2, s_3, \ldots\} = \mathrm{L}(G')$. Then $\mathrm{amb}[\psi_{\mathcal{C}_k}(\langle s_1, \ldots, s_n \rangle)]$ converges to some finite set $\mathcal{G}$ of k-valued grammars over Σ such that*

(i) *for some $G \in \mathcal{G}$, $\mathrm{L}(G') = \mathrm{L}(G)$,*

(ii) *for all $G \in \mathcal{G}$, $\mathrm{L}(G') \subseteq \mathrm{L}(G)$.*

PROOF. Immediate from Lemma 39, using Lemma 33. ∎

Definition Let χ be as defined in the proof of the case $k = 1$, and let $\varphi_{\mathcal{C}_k}(\langle s_1, \ldots, s_n \rangle) = \chi(\mathrm{amb}[\psi_{\mathcal{C}_k}(\langle s_1, \ldots, s_n \rangle)], n)$.

Theorem 2b *$\varphi_{\mathcal{C}_k}$ identifies $\mathcal{C}_k$ in the limit from positive data.*

PROOF. Just like the proof of Theorem 2a. ∎

5 Discussions

We conclude with some implications of our main theorem.

5.1 Simplicity

Let the *length* $|A|$ of type A be the number of occurrences of primitive types in A, and let the *size* $|G|$ of grammar G be $\sum_{a\in\mathrm{dom}(G)}\sum_{\langle a,A\rangle\in G}|A|$. Note that for rigid grammars G and G', if $\sigma[G]\subseteq\sigma[G']$ for some σ, then $|G|\leq|G'|$, and that the size of a k-valued grammar is always the same as that of a disambiguation of it of the smallest size.

Let $\prec$ be a recursive total ordering of all grammars such that if $G\prec G'$, then $|G|\leq|G'|$. If we base our definition of χ on $\prec$, then it is not difficult to see the following: if $G\in\mathrm{range}(\varphi_{\mathcal{C}_k})$, then G is a grammar of the smallest size in $\{\,G'\in\mathcal{C}_k\mid \mathrm{L}(G')=\mathrm{L}(G)\,\}$. Thus, if we now take $\langle\mathcal{C}_k,\mathrm{L}(\cdot)\rangle$ to be the grammar system, our algorithm $\varphi_{\mathcal{C}_k}$ never outputs an unnecessarily complex grammar, and is *simpleminded* in the sense of Osherson, Stob, and Weinstein (1986, p. 64).[3] This does not contradict their Proposition 4.3.6A (p. 65), which says that any class identified by a simpleminded algorithm contains only finitely many grammars, because this proposition essentially depends on the fact that they have indices for all recursively enumerable languages in their grammar system. Thus, their remark 'if children implement recursive, simpleminded learning functions, and if they can only learn languages for which they can produce grammars, then there are only finitely many natural languages' (p. 65) should be taken with a grain of salt. The point has been amply demonstrated by earlier works (see e.g., Angluin 1979, 1982); our result simply reinforces it with reference to a more complex grammar system.

5.2 Identification from Disambiguated Strings

So far, we have assumed that we are given a fixed finite alphabet. In fact, however, our algorithm can work on an infinite alphabet, as long as each grammar (language) is based on a finite subalphabet of it. Let Σ be a fixed finite alphabet, and let $\Upsilon^\infty=\bigcup\{\,\{\,a_i\mid i\in\omega\,\}\mid a\in\Sigma\,\}$. Assume that Υ^∞ is coded in some finite alphabet, by, for example, regarding a_i as a followed by i strokes. Let $\mathcal{C}_1^\infty$ be the class of all rigid grammars over some finite subalphabet of Υ^∞. Then it is clear that essentially the same algorithm as $\psi_{\mathcal{C}_1}$ identifies $\mathcal{C}_1^\infty$ in the limit from positive data.

This fact has a rather interesting consequence. Consider the class $\mathcal{C}$ of all classical categorial grammars over Σ. Then $\mathcal{C}_1^\infty$ is precisely the class of disambiguations of grammars in $\mathcal{C}$. That $\mathcal{C}_1^\infty$ is identifiable in the limit from positive data means that $\mathcal{C}$ is identifiable in the limit from data consisting of *disambiguated*

[3] To be precise, we have to identify grammars that are alphabetic variants in order to make $|G|$ a *size measure* (Osherson, Stob, and Weinstein 1986, p. 63).

strings, that is, strings in the language generated by a certain disambiguation of the target grammar. Disambiguated strings can be regarded as positive data about the target language augmented with certain 'intensional' information about the target grammar. With this additional information, the entire class of context-free languages becomes learnable from positive data with respect to the grammar system of classical categorial grammar.

To be more precise, let us fix a particular mapping from arbitrary classical categorial grammars to their disambiguations. Then, in the terminology of section 2.1, the function from an arbitrary classical categorial grammar G to the language $\mathrm{L}(H)$ generated by its disambiguation H constitutes a naming function. Then the result here is that under this naming function, the class $\mathcal{C}$ of all classical categorial grammars is identifiable in the limit from positive data.

This should be compared with Sakakibara's (1992) result mentioned earlier in section 3.3.3. In his words (p. 59), 'the assumption of examples in the form of structural descriptions strongly compensates for the lack of explicit negative information in positive samples and is helpful for efficient learning of context-free grammars.' While his algorithm ensures an efficient learning of the class of all context-free languages,[4] the assumption that the learner is presented with the skeletal phrase structure of each string that she encounters is probably too strong as a model of first language acquisition. The assumption that the learner can distinguish between different uses of a lexically ambiguous symbol would seem more realistic.[5] Moreover, structures assigned by a reversible context-free grammar are sometimes rather unnatural, and they are rich enough that they can in effect encode information about lexical ambiguity.

Interestingly, F-structures are not sufficient to overcome the lack of negative information. If we take $\mathrm{FL}(\cdot)$ to be the naming function, $\mathcal{C}$ is not identifiable in the limit from positive data (Gold's theorem applies here).

Appendix: Reduction to Shinohara's Theorem

As was mentioned in section 1, Shinohara (1990a, 1990b) proves that the class of context-sensitive grammars with no more than k rules is identifiable in the limit from positive data, for any k. This is done by showing that the class of languages generated by context-sensitive grammars with no more than k rules has the property Wright (1989) called *finite elasticity* (see also Motoki, Shinohara, and Wright 1991). Wright proved that if a class $\mathcal{G}$ of grammars is effectively enumerable and the (universal) membership problem for the grammars in the class is decidable,

[4]The precise complexity of the algorithm $\varphi_{\mathcal{C}_1}$ is yet to be determined.

[5]Note that the notions of 'structure' and 'lexical ambiguity' are both intensional and dependent on the specific grammar that generates the given language. In particular, a rigid classical categorial grammar may have to assign more than one type to a certain symbol a while an equivalent Chomsky normal form context-free grammar may have just one rule of the form $A \Rightarrow a$.

finite elasticity of the class of languages generated by the grammars in $\mathcal{G}$ guarantees identifiability of $\mathcal{G}$ from positive data. Finite elasticity is an extensional property of classes of languages, and it is closed under subclasses. So Shinohara's result implies that the class of languages generated by ϵ-free context-free grammars with no more than k rules has finite elasticity. Given this, we obtain an alternative proof of our main theorem by the following observation:

Proposition 41 *For any k-valued classical categorial grammar over Σ, there is an equivalent ϵ-free context-free grammar with no more than $2k|\Sigma| - 1$ rules.*

PROOF. First, note that every k-valued grammar G has an adequate k-valued grammar G' such that $\mathrm{FL}(G) = \mathrm{FL}(G')$. To see this, take a disambiguation H of G, find an adequate rigid grammar H' such that $\mathrm{FL}(H) = \mathrm{FL}(H')$ by Proposition 11, and let $G' = \mathrm{amb}[H']$.

So take an adequate k-valued grammar G. Assume that $G \neq \emptyset$. Construct a Chomsky normal form context-free grammar $G_0 = \langle \Sigma, \mathrm{SubTp}(G), t, P_0 \rangle$ as follows:

- The set of non-terminals of G_0 is $\mathrm{SubTp}(G)$.
- The start symbol of G_0 is t.
- The set P_0 of rules of G_0 is $\{ B \to B/A\ A \mid B/A \in \mathrm{SubTp}(G) \} \cup \{ B \to A\ A\backslash B \mid A\backslash B \in \mathrm{SubTp}(G) \} \cup \{ A \to a \mid \langle A, a \rangle \in G \}$.

Obviously, $\mathrm{L}(G_0) = \mathrm{L}(G)$, where $\mathrm{L}(G_0)$ is defined in the standard way for context-free grammars. Since G is adequate, G_0 is a context-free grammar with no useless symbol.

Let $\mathcal{A} = \{ A \in \mathrm{SubTp}(G) \mid$ there is only one rule in G_0 whose left-hand side is $A \}$. For $A \in \mathcal{A}$, let $\mathrm{right}(A)$ be such that $A \to \mathrm{right}(A)$ is a rule in P_0. Let $P_1 = P_0 - \{ A \to \mathrm{right}(A) \mid A \in \mathcal{A} - \{t\} \}$. Now eliminate all occurrences of non-terminals in $\mathcal{A}$ on the right-hand side of rules in P_1 by repeatedly replacing such occurrences of $A \in \mathcal{A}$ by $\mathrm{right}(A)$. This process must terminate, for, if it does not, there must be a cycle of non-terminals $A_0, A_1, \ldots, A_n = A_0$ $(n \geq 1)$ such that each A_i $(0 \leq i \leq n)$ is in $\mathcal{A}$ and A_{i+1} occurs in $\mathrm{right}(A_i)$ for $0 \leq i \leq n-1$, which implies that $A_0, A_1, \ldots, A_n$ are useless symbols in G_0. Let P_2 be the result of applying this process to P_1. Let $G_2 = \langle \Sigma, \mathrm{SubTp}(G) - (\mathcal{A} - \{t\}), t, P_2 \rangle$. It should be clear that $\mathrm{L}(G_2) = \mathrm{L}(G_0)$.

It remains to show that $|P_2| \leq 2k|\Sigma| - 1$. This can be seen as follows. Define a binary relation $\sqsubset$ on $\mathrm{SubTp}(G) \cup G$ as follows: $\sqsubset = \{ \langle B, B/A \rangle \mid B/A \in \mathrm{SubTp}(G) \} \cup \{ \langle B, A\backslash B \rangle \mid A\backslash B \in \mathrm{SubTp}(G) \} \cup \{ \langle A, \langle a, A \rangle \rangle \mid \langle a, A \rangle \in G \}$. There is a one-to-one correspondence between the pairs in $\sqsubset$ and the rules in P_0. The graph of $\sqsubset$ consists of m rooted trees, where $m = |\mathrm{Pr} \cap \mathrm{SubTp}(G)|$. The number of leaf nodes of these trees is at most $k|\Sigma|$, and it is not difficult to see that the number of nodes in these trees which have more than one daughter is at most $k|\Sigma| - m$. Then, the number of pairs $\langle A, X \rangle \in \sqsubset$ such that A has more than

one daughter in the graph of $\sqsubset$ is at most $k|\Sigma| + k|\Sigma| - m - m = 2k|\Sigma| - 2m$, since those correspond one-to-one to either leaf nodes or nodes with more than one daughter that are not highest. Thus, $|P_2| \leq 2k|\Sigma| - 2m + 1 \leq 2k|\Sigma| - 1$.[6] ■

References

Angluin, D., 1979, Finding patterns common to a set of strings, in *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, 130–141.

Angluin, D., 1980, Inductive inference of formal languages from positive data, *Information and Control* **45**, 117–135.

Angluin, D., 1982, Inference of reversible languages, *Journal for the Association for Computing Machinery* **29**, 741–765.

Buszkowski, W., 1987, Discovery procedures for categorial grammars, in E. Klein and J. van Benthem (eds.), *Categories, Polymorphism and Unification*, University of Amsterdam, Amsterdam.

Buszkowski, W., 1988, Generative power of categorial grammars, in R. Oehrle, E. Bach, and D. Wheeler (eds.), *Categorial Grammars and Natural Language Structures*, D. Reidel, Dordrecht.

Buszkowski, W. and G. Penn, 1990, Categorial grammars determined from linguistic data by unification, *Studia Logica* **49**, 431–454.

Gold, E. M., 1967, Language identification in the limit, *Information and Control* **10**, 447–474.

Motoki, T., T. Shinohara, and K. Wright, 1991, The correct definition of finite elasticity: Corrigendum to identification of unions, in *The Fourth Annual Workshop on Computational Learning Theory*, p. 375, Morgan Kaufmann, San Mateo, CA.

Osherson, D., M. Stob, and S. Weinstein, 1986, *Systems That Learn*, MIT Press, Cambridge, MA.

Sakakibara, Y., 1992, Efficient learning of context-free grammars from positive structural examples, *Information and Computation* **97**, 23–60.

Shinohara, T., 1990a, Inductive inference from positive data is powerful, in *The 1990 Workshop on Computational Learning Theory*, pp. 97–110, Morgan Kaufmann, San Mateo, CA.

Shinohara, T., 1990b, Inductive inference of monotonic formal systems from positive data, in S. Arikawa, S. Goto, S. Ohsuga, and T. Yokomori (eds.), *Algorithmic Learning Theory*, pp. 339–351, Ohmsha, Tokyo, and Springer, New York and Berlin.

Wexler, K. and P. Culicover, 1980, *Formal Principles of Language Acquisition*, MIT Press, Cambridge, MA.

[6]Equality holds when $k = |\Sigma| = 1$. Otherwise it can be shown that $|P_2| \leq 2k|\Sigma| - 2$.

Wright, K., 1989, Identification of unions of languages drawn from an identifiable class, in *The 1989 Workshop on Computational Learning Theory*, pp. 328–333, Morgan Kaufmann, San Mateo, CA.