Formal Properties of Categorial Grammars

Gabriel D. Carroll

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Arts with Honors

Department of Mathematics
and Department of Linguistics

Harvard University

May 31, 2005

Abstract

We discuss two standard formal tools used to study models of grammar. One of these is formal language theory, which provides a way to describe the complexity of languages in terms of a sequence of standard language classes known as the Chomsky hierarchy. The other tool is learnability theory, which can describe, for a given class of languages, whether or not there exists a single learner that can learn every language in the class; we use a particular model for learning developed by Gold. These two tools can be used to obtain formal properties of a grammar system, and to evaluate the validity of a theory of natural language. After presenting the tools, we show how they can be applied to the linguistic theory of categorial grammars, and we discuss the results.

1 Introduction

Suppose you are a linguist, with a theory of the human language faculty — that is, a system of principles that makes some predictions (not necessarily comprehensive) about how language works — and you would like to evaluate this theory. Or perhaps you have several different theories, and you would like to compare them. How do you go about doing this?

The most basic approach you employ is to see whether the theory can account for empirical data about what sentences are grammatical or ungrammatical in particular languages, and whether it provides reasonable ways of accounting for differences between languages. Depending on the nature of the theory and the predictions it makes, you may have other tools available: psycholinguistic experimentation, for example, or evidence from child language acquisition, or theoretical considerations of parsimony. The purpose of the present paper is to present some of the purely formal tools that are also of use in evaluating a theory of language.

The general setup is as follows: The cognitive faculty known as *universal grammar*, which is shared by all humans and provides the basis for acquisition and use of languages, delimits some set $\mathcal{L}^*$ of possible human languages; the exact nature of this set is not known to us. Any linguistic theory predicts some class $\mathcal{L}$ of possible human languages (or at least makes some predictions about $\mathcal{L}$, even if it does not completely determine the class). The theory is a good one insofar as $\mathcal{L}$ is a good approximation to $\mathcal{L}^*$. Therefore, if there are certain formal properties that are believed, for independent reasons, to hold for $\mathcal{L}^*$, then we can evaluate the theory by determining whether or not its $\mathcal{L}$ also has these properties.

We will describe two of the formal tools along these lines — formal complexity and learnability theory — and will apply them to a particular theory of the syntactic component of language, known as *categorial grammar* (see e.g. [38], [68], [77]). More precisely, there have been various different versions of categorial grammar proposed in the linguistic literature (classical categorial grammar, combinatory categorial grammar, and so forth). We define a sort of "meta-grammatical" framework, within which we can define specific theories of categorial grammar; one would then add a lexicon in order to obtain a grammar for a specific language. We then use these formal tools to compare some of the various forms of categorial grammar that can be obtained by instantiating this framework. Although the questions we are studying are too broad to obtain anything like an exhaustive analysis, we can at least hope to have demonstrated that the tools themselves are interesting and merit further development, and that the concepts we have introduced might be of use in future investigation of these questions.

In the direction of formal complexity, we will show that some versions of categorial grammar are inadequate for describing natural language, others are excessively general and allow for languages that are more complex than any human language is believed to be, while others seem to strike a reasonable middle ground. In the direction of learnability, under the particular model we use (the Gold framework of learning from text, [31]), we will show that, while even the most basic version of the theory gives a set $\mathcal{L}$ of languages that is too large to be learned by any one learner (and is therefore unrealistic), imposing a simple restriction known as "rigidity" on a categorial grammar system results in a class of languages that can be learned, and in a very elegant and natural manner.

Although there are some new concepts and new results, the main purpose of the present work is expository. We are interested in presenting tools at the intersection of mathematics and linguistics, for the benefit of the reader in either discipline who may be unfamiliar with them; surely the insight that is gained when scholars are exposed to ideas from outside their field is justification enough for the endeavor. In particular, we seek to show that the application of mathematical methods can provide concrete and meaningful results of interest in linguistics. Another goal, in the course of the exposition, is to develop the relevant definitions and results from the ground up; currently, essentially all of the non-book-length literature in this field is either vague and fuzzy or extremely technical. For practical reasons, the writer must choose his audience, and so this paper is directed primarily at mathematicians with no background in linguistics, rather than vice versa. Consequently, it will be necessary to spend some time on basic notions that will be tediously familiar to the linguist (including much of this introduction), whereas standard concepts from set theory and algebra will be used without comment. Should the latter cause any trouble, the reader is generally referred to [49] for basic mathematical ideas that are used in linguistics; a few of our mathematical examples use terms not found in that book, but they will not be crucial to the exposition here.

In view of the impossibility of appealing equally to every reader, the least we can do is to provide an outline to guide the reader to the sections he or she will care the most about. In Section 2, we lay out the tools of formal language theory for describing the complexity of a class of languages, and we discuss their relevance to the complexity of the natural languages ($\mathcal{L}^*$); although the ideas of the first few pages are essential, the reader who is less interested in these questions can skim the rest of the section. In Section 3, we describe the framework of categorial grammars. Much of the section is devoted to examples from human languages and can be skimmed by the mathematical reader interested only in the formalism instead of the motivation; the crucial definitions are at the end of the section. Section 4 applies the ideas of formal language theory to categorial grammars and produces general results about the complexities of various versions of the categorial grammar framework. Section 5 is a brief overview of learnability theory and should be relatively interesting for everyone. In Section 6, we apply learnability to categorial grammars; the ideas of this section probably have the most inherent interest to the mathematician, although much of it is somewhat technical, and some readers may want to focus on results and skip the proofs. Although the mathematical theory here is more interesting, the results are less general than those of Section 4, and we close the section with a discussion of some of their shortcomings. Section 7 summarizes our results and discusses their overall relevance. Finally, in some places, including a proof of a particular result would needlessly interrupt the discussion; these proofs are included in the appendix. We also include, after the references, an index listing the most important terms we introduce and the pages on which they are defined, for when the jargon becomes overwhelming.

Having now provided the requisite general orientation, we will spend a bit more time discussing the methodological and philosophical basis of our approach. We make a number of basic assumptions that are common to the generative grammar tradition (e.g. [16, ch. 1]), and these assumptions (and the reasons for them) had better be on the table in order for our work to make any sense. Some of the ideas are elaborated in more detail in the popular treatment [53].

To have a description of a language, we must have a "grammar" that encapsulates sufficient information to determine what strings are and are not grammatical sentences of the language. The most basic maxim is that languages' grammars are not completely independent of each other but rather are built on a foundation of "universal grammar" that captures the similarities across all human languages. Some of these similarities are so basic as to escape everyday notice: for example, the expressions of a language are temporally ordered strings, not unordered sets or three-dimensional arrays; these strings contain hierarchically structured units, such as sounds, morphemes, words, phrases, and sentences. Some are less blindingly obvious but still require expression and explanation: no language has a rule where one says "Cow!" at the end of a sentence to indicate that the subject is gray; every language has nouns, verbs, and adjectives. These commonalities comprise universal grammar. Universal grammar then provides a framework for specifying the grammar of a particular language; such a grammar is now a finite system of rules or parameter values. The rules of universal grammar, together with the details of a grammar for a language L, define L. Thus, universal grammar demarcates a set $\mathcal{L}^*$ of possible human languages L.

One of the major guiding goals of modern linguistics is to understand universal grammar, as an abstract description of language, in the same way that, say, the Standard Model is a description of the physical universe. Various theories of universal grammar are out there in the literature. As already described, each theory predicts a class $\mathcal{L}$ of possible languages, which is intended to approximate the actual class $\mathcal{L}^*$. We apply formal tools to these theories (in our case, categorial grammar) to see if $\mathcal{L}$ has certain properties that are known or believed to hold for $\mathcal{L}^*$.

Formal language theory ([33], [59]) is an active area on the border of mathematics and computer science, although much of the founding work in the area actually grew out of linguistics. Among other things, it provides a way of talking about desirable properties a class of languages should have and defines several canonical classes that form a hierarchy of complexity levels. These provide a yardstick for making statements about the complexity of languages, and in particular one can make non-vacuous statements about the complexity of human languages. From empirical data about acceptable and unacceptable sentences, one can establish lower bounds on the complexity of languages in $\mathcal{L}^*$ in terms of this hierarchy. Upper bounds are harder to talk about with confidence, but there are reasons for us not to ascribe any higher complexity to $\mathcal{L}^*$ than we have to: "most" of the constructions of natural languages can be accounted for by context-free languages (this being one of the levels of the standard hierarchy); there are particular structural properties that human languages seem to have that languages at higher complexity levels seem to lack; on the non-empirical side, the human brain must be able to parse utterances reasonably efficiently, and more complex languages potentially produce parsing problems of greater time and space complexity. Hence, formal language theory gives us means by which to compare $\mathcal{L}$ and $\mathcal{L}^*$. We should be wary of a theory of language that predicts higher levels of complexity for languages in $\mathcal{L}$ than are known to be needed for human language. We should also reject (or revise) a theory that restricts $\mathcal{L}$ to low levels that are known to be inadequate.

Learning and learnability theory, an area that today receives attention predominantly in artificial intelligence, also provides a way to reject theories of language that predict an excessively large $\mathcal{L}$. Any normal child, in its period of language acquisition, can learn any possible human language. Therefore, $\mathcal{L}^*$ must be small enough that there exists a single learning mechanism capable of learning every language $L \in \mathcal{L}^*$. Thus, we have a property of $\mathcal{L}^*$ that holds on purely theoretical grounds, without even any need for data to support it. We then have a test for theories of language: if a theory predicts an $\mathcal{L}$ so large that no one mechanism can learn every language in the class, then the theory does not adequately delimit the space of human languages. Of course, it remains to fill in exactly what it means to learn a language, what kind of learning mechanism is allowed, the form of its input and output, and so forth; numerous models for learning exist in the literature. We use one of the simplest models, basically for reasons of tractability rather than empirical validity, but it will still give us interesting and nontrivial results.

Linguistics is not currently at a stage where anyone seriously claims to have a comprehensive theory of universal grammar. The use of these tools, then, is to compare the various approximations and partial theories that research has thus far discovered, and determine which of the various paths that have been explored are most promising for future understanding of language. And what, one might ask, happens next? In fact, understanding language is not only an end in itself but also a window into understanding of the human mind. Basic developments in linguistics have already contributed to major insights in other areas of cognitive science (see [15], [28], [34]).

Indeed, universal grammar cannot exist in an abstract vacuum but must be reflected, in some form, in the human brain. The hypothesis that the brain is hard-wired with a framework for language is necessary to explain many of the universal commonalities among all languages; it is also necessary to explain how it is possible for the human child to acquire language, as the data available to the child are grossly inadequate to learn a language if no information is available a priori (this is the so-called *logical problem of language acquisition*, [6], [21], [36]).

It is important to keep in mind, however, that the grammar is not the same thing as the neural system underlying language. A grammar is a passive system of rules determining well-formedness of sentences, whereas the neural system (the *computation*) must operate temporally to create and interpret sentences; the neural system produces only an approximation of the language given by the grammar (see below on the *competence-performance* distinction); and even if this were not the case, we should pursue the descriptive goal of constructing grammars because it is more accessible than the goal of neurologically understanding how sentences are produced and understood. Chomsky, in [18, ch. 5], compares a grammar to a theory of chemistry that describes what chemical compounds are structurally possible; such a theory does not directly give processes for synthesizing these compounds, but it could provide the basis for such processes. Similarly, we study grammars in order to provide the basis for future understanding of the cognitive process; although the two are not identical, there is surely some relation between them.

Since it must be represented in the mind, the language faculty is basically a biological mechanism. If one accepts the argument that language has been shaped by adaptive pressures over the course of evolution (see [52]), then the broad questions we are asking will help us understand not only how language works, but in some sense *why* it works as it does. Among adaptive constraints, there is a tension between pressures toward communicative versatility and flexibility (which tend to make the set $\mathcal{L}^*$ large and the languages in it complex), and considerations of parsing speed and learnability (which make $\mathcal{L}^*$ small and simple). In the course of establishing a theory of universal grammar, these considerations may help us understand not only whether our theory is correct but why our universal grammar came to be this way and not some other way. Of course, we should not exaggerate the depth of evolutionary understanding that is available at present. Much of evolution is random, and constraints may arise from outside language proper. However, it is the task of linguistics to explain as much of language as possible without recourse to external forces, and the task of other sciences to contribute the external parts of the explanation.

Having finished digressing to explain the long-term goals of understanding language, we should continue to state the methodological assumptions that we make. One basic assumption is the *competence-performance* distinction. Linguistic *competence* refers to the subconscious knowledge possessed by an idealized speaker, knowledge that is necessary in order to produce and understand utterances correctly. *Performance* is the speaker's (or hearer's) use of that knowledge, which is subject to limitations of memory or time, distractions, physical constraints, and so forth. The grammars we study are objects of competence, not of performance. Thus, for example, even though there are only finitely many sentences a person can ever physically utter (because an excessively long sentence would cause the speaker to starve to death), a competence grammar is in theory capable of generating infinitely many sentences. Conversely, because of stuttering, false starts, or sudden shifts of attention, a person may produce utterances which we do not want to consider grammatical.

We study competence, but the only data empirically available to us are filtered through performance. Noisy data is a property of any empirical science, and although some data can be considered faulty for this reason, we have more than enough robust data to formulate interesting questions that any theory of competence should be able to answer. The reason we focus on competence here is the same reason that any science makes idealizations: the ideal object is more tractable than the real and directly observable one.

Another assumption we make is that a grammar classifies every string (sentence) as grammatical or ungrammatical. In practice, there are various shades of acceptability. (Note that there is a distinction between "grammaticality," which is determined by the abstract grammar, and "acceptability," which is a performance judgment, [19].) But we use this black-and-white system, again, as an approximation to make the objects of study more tractable. There are plenty of data on sentences that are clearly grammatical or clearly ungrammatical, so that accounting for these data poses interesting problems, and we need not further muddy the waters by allowing ourselves to use data about intermediate levels.

One more assumption that we implicitly make is that children accurately learn the language of their parents or their community. This is clearly not entirely true — if it were, languages would never change over time — but once again is assumed for the sake of simplicity.

Having now spent enough time channeling the voice of Noam Chomsky, we will make some brief remarks about categorial grammar, the theory we plan to study (it will be described more fully in Section 3). Categorial grammar, insofar as we will be concerned with it, is a theory of the syntax of natural language: it makes claims about what sentences are syntactically well-formed or ill-formed; we will have nothing to say about phonology, semantics, discourse structure, or these various other subunits of linguistics. (This means, of course, that we are assuming that sentences can be empirically classified as syntactically grammatical or ungrammatical; we have the intuition that a sentence may make no semantic sense but be syntactically fine, or vice versa. For the data we use, these assumptions are reasonable.) It is important to be explicit about this distinction because it is not always observed in the literature; for example, the discussion of programming languages in [33, ch. 6] or of number names in [56], which give conclusions that are irrelevant because they ascribe to syntax some judgments that properly belong to the domain of semantics or pragmatics.

Categorial grammar ([38],[68],[77]) is somewhat of a dissident tradition that differs in fundamental ways from mainstream work in syntax. Nonetheless, it stands as a well-developed linguistic theory. A major source of its appeal in linguistics is that it provides a tight and clean interface between the syntax and the semantics. It also gives an intuitively believable description of the process of producing sentences, constructing them gradually from the ground up, simultaneously applying syntactic rules to assemble constituents and semantic rules to combine their meanings into a meaning for the longer string. Although, as we have already remarked, the grammar and the computation are distinct entities, there is nonetheless a certain appeal to a form of grammar that naturally lends itself to producing theories about the possible nature of the computation. Indeed, as argued in [10], the minimalist research program ([22]) that has become popular in recent years is actually founded on some of the same principles that drive interest in categorial grammar, leading to some basic similarities between the two grammatical frameworks, even though they are superficially quite dissimilar. All this said, we will not need to concern ourselves further with the reasons that categorial grammar is interesting to linguists and will focus on the formal properties of the system itself.

There is also a more basic reason that we choose categorial grammars for our focus: the general framework is relatively simple and elegant to state, which makes it both suitable for an exposition of the tools of mathematical linguistics and relatively tractable in terms of obtaining some actual results. We do not wish to exaggerate the immediate significance of our results — any real convincing theory of universal grammar would have to be much more detailed and complex than the framework we provide here — but they can at least provide some direction to study, and a foundation for further results about more sophisticated linguistic models.

We should remark that the corpus of literature on categorial grammar is sufficiently broad and diverse that it is impossible for us to give here a fair exposition of the various versions of the theory that have cropped up, let alone a unified formalism to cover all of them (the reader is referred to [77] for such an overview). The framework we will develop is intended to generalize cleanly many of the main ideas that have been developed in different versions of categorial grammar, but inevitably some ideas that have been fruitful in the linguistic theory will have to be shortchanged. To those whose relevant ideas have been omitted here: sorry about that, better luck next time.

2 Basic formal language theory

In this section, we introduce the basics of formal language theory and discuss its applications to natural language. Formal language theory is a well-developed field in its own right, with connections to programming languages, logic, and even biology. The material here, except for a few of the applications we mention, is entirely standard; see [33] or [59] for more detailed exposition of the theory.

Given a (generally finite) set of symbols Σ, let Σ^* be the set of all finite sequences of symbols from Σ. This is a monoid under the operation of concatenation; the identity is the empty string, which we refer to as ϵ. If $v, w \in \Sigma^*$, we write vw for their concatenation; v^n is the n-fold concatenation of v. We also let Σ^+ denote the set $\Sigma^* \setminus \{\epsilon\}$ of all nonempty sequences of symbols from Σ. The length of an element $w \in \Sigma^*$ is denoted $|w|$. A *language over* Σ is any subset $L \subseteq \Sigma^*$.

For purely formal examples, we will use Roman letters $\mathrm{a}, \mathrm{b}, \mathrm{c}, \ldots$ to denote elements of Σ. (Note that we use non-italic letters as specific terminal symbols; italics such as a will be variables ranging over Σ.) In linguistic applications, we think of Σ as some lexicon of words or morphemes, and L as the set of grammatical sentences of the language. The elements of Σ may be thought of as abstract lexical entries rather than phonological forms. We will refer to sequences from Σ^* as *sentences* or *strings* and elements of Σ as *terminals*. Σ is sometimes also called the *vocabulary*. Notice again that we are making the idealizing assumption that a language classifies every string as being either a grammatical sentence (i.e. in L) or not. Also, we are considering a language to be only a set of strings, ignoring (for now) their internal structure and meaning; we will return to this point at the end of the section.

In general, a language is an infinite set. Informally speaking, the term *grammar* is used to refer to a finite "recipe" that describes a language, and a *grammar system* is a function that maps grammars into languages. We say that a grammar *generates* a language; sometimes we also speak of the grammar *generating* each individual string in the language. Similarly, we can speak of the grammar system as *generating* a class of possible languages (or as generating each language in the class). The class of languages that can be generated by grammars within a given system is called the *generative capacity* or *expressive power* of the system.

Here is an example. Suppose we have a finite set V of symbols, which we call *nonterminals*, disjoint from Σ, and a finite set P of *productions* (or *rules*), which are ordered pairs (v, w) where $v, w \in (V \cup \Sigma)^*$. A production is normally notated as $v \to w$. For $x, y \in (V \cup \Sigma)^*$, we write $x \Rightarrow y$ if there exist $s, t, v, w \in (\Sigma \cup V)^*$ such that $x = svt, y = swt$, and $v \to w$ is a production. We write $x \overset{*}{\Rightarrow} y$ if there exists a sequence $x = x_0, x_1, \ldots, x_n = y$ such that $x_i \Rightarrow x_{i+1}$ for each i. Thus, $x \overset{*}{\Rightarrow} y$ if we can obtain y from x by successively rewriting substrings according to the productions in P. Such a sequence $x_0, \ldots, x_n$ is called a *derivation* of y from x.

If $S \in V$ is a distinguished "start" symbol, then the triple (V, S, P) is a grammar, called an *unrestricted rewrite system* over the terminal alphabet Σ. We define the *rewrite language* generated by this grammar to be the set

$$L = \{x \in \Sigma^* \mid S \Rightarrow x\}.$$

Thus, the language is the set of all strings of terminals that can be obtained from S by successively applying the productions of P. (This explains the names "terminal" and "nonterminal" — in the process of applying productions, we have arrived at a sentence of L if there are no nonterminals left.)

For example, if $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, V is just $\{S\}$, and P consists of the productions

$$S \to \mathrm{a}S\mathrm{bc} \qquad S \to \epsilon \qquad \mathrm{ab} \to \mathrm{ba} \qquad \mathrm{ac} \to \mathrm{ca} \qquad \mathrm{bc} \to \mathrm{cb} \qquad \mathrm{cb} \to \mathrm{bc}$$

then the corresponding rewrite language consists of all terminal strings containing an equal number of a's, b's, and c's, as is easy to check. P also generates strings such as aa$S$bccb, but this is not in the language because it contains the nonterminal S.

Arbitrary rewrite languages can be extremely complex, so one generally studies more restricted classes of rewriting systems. For example, (V, S, P) is called a *right linear grammar* if every rule in P is of the form

$$X \to aY \qquad \text{or} \qquad X \to \epsilon \tag{1}$$

with $X, Y \in V$, $a \in \Sigma$. Languages generated by right linear grammars are called *regular* languages.

Regular languages come up in many places in mathematics. The set of strings containing the substring abb is a regular language (take nonterminals S, T, U, V, and the rules $S \to \mathrm{a}T$, $T \to \mathrm{b}U$, $U \to \mathrm{b}V$, $V \to \epsilon$, as well as $S \to aS$, $V \to aV$, for all $a \in \Sigma$). More generally, the set of strings containing any given substring is regular, as is the set of strings not containing any given substring. If Σ is a generating set for a finite group H, then the set of strings $a_1 \cdots a_n$, where the product $a_1 \cdots a_n$ equals the identity of H, is regular (just have one nonterminal X_h for each $h \in H$, and have the rules $S \to hX_h$, $X_g \to hX_{gh}$, $X_e \to \epsilon$). If natural numbers in decimal representation are viewed as strings over $\Sigma = \{0, 1, \ldots, 9\}$, the set of multiples of 23 is a regular set.

One can also try to use regular languages to generate sentences of human languages. For example, consider the following grammar (where the English words represent terminals and uppercase letters are nonterminals):

$$S \to \textit{the } T \qquad S \to \textit{a } T$$

$$T \to \textit{big } T \qquad T \to \textit{sleepy } T \qquad T \to \textit{red } T$$

$$T \to \textit{boy } U \qquad T \to \textit{girl } U \qquad T \to \textit{raccoon } U$$

$$U \to \textit{walks } V \qquad U \to \textit{reads } V \qquad U \to \textit{thinks } S \qquad V \to \epsilon$$

This gives us derivations such as

$$S \Rightarrow \textit{the } T \Rightarrow \textit{the sleepy } T \Rightarrow \textit{the sleepy girl } U \Rightarrow \textit{the sleepy girl reads } V$$

$$\Rightarrow \textit{the sleepy girl reads}$$

or (abbreviating a few steps now)

$$S \Rightarrow \textit{a } T \Rightarrow \textit{a boy } U \Rightarrow \textit{a boy thinks } S \overset{*}{\Rightarrow} \textit{a boy thinks a raccoon walks.}$$

As we shall see, there are convincing reasons why this is not an adequate model of natural language.

Regular languages are relatively simple-minded creatures. The following result shows one way in which they are simple:

Lemma 1 *(Iteration Theorem) If L is a regular language, there exists a constant k with the following property: whenever $x \in L$ and $|x| > k$, then there exists a factorization $x = uvw$, with $0 < |v| \leq k$, such that $uv^nw \in L$ for all $n \geq 0$.*

Proof: Let (V, S, P) be a right linear grammar for L, and put $k = |V|$. If $x \in L$, consider the derivation

$$S = x_0 \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_n = x.$$

We readily see by induction that x_i consists of i terminals followed by a nonterminal, for each $i < n$, and x_n is just a string of $n - 1$ terminals. If $n - 1 > k$, then some two steps of the derivation, that are at most k steps apart, must end in the same nonterminal X. It follows that our derivation can be broken into subderivations of the form

$$S \overset{*}{\Rightarrow} uX \overset{*}{\Rightarrow} uvX \overset{*}{\Rightarrow} uvw = x$$

for some strings $u, v, w \in \Sigma^*$, where $0 < |v| \leq k$, and in particular the middle subderivation tells us that $X \overset{*}{\Rightarrow} vX$. Then we have the derivation

$$S \overset{*}{\Rightarrow} uX \overset{*}{\Rightarrow} uvX \overset{*}{\Rightarrow} uvvX \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} uv^nX \overset{*}{\Rightarrow} uv^nw$$

for any n, so $uv^nw \in L$. □

So, for example, while $\{\mathrm{a}^n\mathrm{b}^m \mid n, m \geq 0\}$ is a regular language (consider the productions $S \to \epsilon, S \to \mathrm{a}S, S \to \mathrm{a}T, S \to \mathrm{b}T, T \to \mathrm{b}T, T \to \epsilon$), one finds by an easy application of the preceding theorem that $\{\mathrm{a}^n\mathrm{b}^n\}$ is not a regular language. (The subword v, if it exists, either consists entirely of all a's, consists entirely of b's, or crosses the a-b border; we readily check that each of these cases gives a contradiction.) Informally speaking, the iteration theorem tells us that the strings of a regular language L cannot be governed by a constraint that requires some relationship between arbitrarily long substrings.

A more general class of languages is given by the *context-free grammars*. These are rewriting systems in which every rule is of the form

$$A \to w \qquad \text{for some } A \in V, w \in (V \cup \Sigma)^*. \tag{2}$$

Thus we impose only the condition that every rule should rewrite a single nonterminal. The languages generated by these grammars are *context-free languages*. Clearly every regular language is context-free, but not conversely; for example, the productions $S \to \mathrm{a}S\mathrm{b}, S \to \epsilon$ give us that $\{\mathrm{a}^n\mathrm{b}^n \mid n \geq 0\}$ is a context-free language.

Context-free languages are worth dwelling on, because they have long been used to model natural language; what have classically been called "phrase-structure grammars" are context-free grammars in our parlance. The idea is that the nonterminals denote grammatical categories (this corresponds to the lay term "parts of speech," not to be confused with the usual mathematical sense of "categories"), such as sentence, noun, noun phrase, verb phrase, determiner, and so forth. A grammar would consist of rules specifying how categories combine to form other categories, such as

$$\begin{aligned} \mathrm{S} &\to \mathrm{NP\ VP} \\ \mathrm{VP} &\to \mathrm{V\ NP} \\ \mathrm{NP} &\to \mathrm{D\ N} \end{aligned}$$

and rules assigning individual lexical items (terminals) to categories, such as

$$\mathrm{V} \to \textit{ate} \qquad \mathrm{N} \to \textit{cake} \qquad \mathrm{N} \to \textit{crocodile} \qquad \mathrm{D} \to \textit{a} \qquad \mathrm{D} \to \textit{the}$$

(Note that we will use non-italicized letters to represent grammatical categories in natural-language examples.)

We then get a derivation for an English sentence such as *the crocodile ate the cake*:

$$\begin{aligned}\mathrm{S} \Rightarrow \mathrm{NP\ VP} \Rightarrow \mathrm{D\ N\ VP} &\Rightarrow \mathrm{D\ N\ V\ NP} \Rightarrow \mathrm{D\ N\ V\ D\ N} \\ &\overset{*}{\Rightarrow} \textit{the crocodile ate the cake} \qquad (3)\end{aligned}$$

For succinctness, we have not explicitly shown the last five steps (in which the terminals are substituted for $\mathrm{D, V, N}$, one at a time). Similarly, *a crocodile ate the cake*, *the cake ate a cake*, *a cake ate the crocodile* and so forth are all sentences in the language.

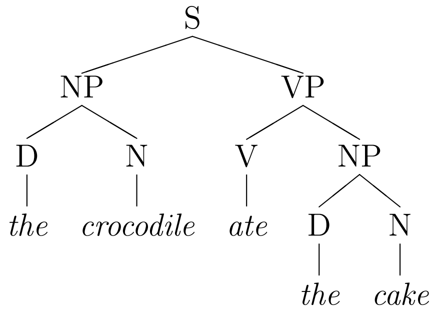
The language described above contains only finitely many strings, but as with our regular-language example, we can spice things up; just add a couple rules:

$$\mathrm{VP} \rightarrow \mathrm{V\ S} \qquad \mathrm{V} \rightarrow \textit{thinks}$$

Now notice that we have the derivation S ⇒ NP VP ⇒ NP V S ⇒ NP *thinks* S, which gives us embedded sentences. Hence, our grammar now generates sentences such as *the crocodile thinks a crocodile ate the cake*, *a cake thinks the crocodile thinks a cake ate a cake*, and so forth. Unfortunately, we also get misfits such as *a cake ate the crocodile thinks a cake* sneaking into our language. We could rectify this by having a new nonterminal for verbs that take a sentence as complement, or more simply by replacing our two newest rules with VP → *thinks* S. But for now, we have developed this example far enough.

Context-free languages are motivated primarily by linguistic examples like the above, but they also come up independently in mathematics (although not as often as regular languages). For example, one frequently occurring language in combinatorics is the semi-Dyck language over $\{\mathrm{a}, \mathrm{b}\}$ ([33, ch. 10]), consisting of all strings containing equal numbers of a's and b's such that no initial segment contains more b's than a's. (It may be more familiar as the set of sequences of left and right parentheses that match properly.) This language is generated by the context-free grammar with rules $S \rightarrow SS$, $S \rightarrow \mathrm{a}S\mathrm{b}$, $S \rightarrow \epsilon$. Over a free group, with Σ the set of generators and their inverses, the set of strings $a_1 \cdots a_n$ where the product equals the identity is a context-free language (generated by $S \rightarrow SS$, $S \rightarrow aSa^{-1}$ for $a \in \Sigma$, and $S \rightarrow \epsilon$). In fact, it has been shown that a group has the property that the "cancellative words" form a context-free language if and only if it has a free subgroup of finite index ([47]). Context-free grammars can also be used in logic, to describe well-formed formulas; for example, one can describe well-formed arithmetical expressions by rules such as $S \rightarrow (S+S), S \rightarrow (S \cdot S), S \rightarrow a$ (where a can be any variable and $(,), +, \cdot$ are terminals).

Suppose we have a fixed context-free grammar. To any derivation of a string from a single nonterminal, we can associate a *parse tree*; this is an ordered tree whose leaves are labeled by terminals (or ϵ) and whose intermediate nodes are labeled by nonterminals, so that for each intermediate node labeled X, the children (in order) spell out some w such that $X \rightarrow w$ is a production used in the derivation. For example, the derivation for (3) above gives us the following tree:



(4)

In general, for each sentence $w \in L$, we have at least one parse tree whose root is labeled S and whose leaf labels, in order, spell out w. (A parse tree whose root is labeled with the start symbol will be called an *S-parse tree*.) Notice that a parse tree does not, in general, uniquely determine a derivation: for example, to recover a derivation from the above tree, we must first use the production S → NP VP, but then we have the choice as to which of the rules NP → D N, VP → V NP to apply next.

In showing how context-free grammars (or rewriting systems more generally) can be used to model natural language, it should be mentioned that the symbol → is merely a notational device; it does not describe any temporal process by which sentences are constructed. Thus, the derivations above should not be taken to mean that a human speaker begins with the symbol S, decomposes it into other nonterminals, and finally inserts words; indeed, we would expect the process to be the opposite — the speaker starts with the words and gradually assembles them into larger constituents. Remember, as discussed in Section 1, that the grammar is not the same thing as the computational process that produces sentences.

We have seen that the context-free languages are a proper superset of the regular languages. However, the context-free languages satisfy an analogue of the Iteration Theorem, giving a tight constraint on their complexity:

Theorem 2 *(Pumping Lemma) If L is a context-free language, then there exists a constant k with the following property: for any $z \in L$ with $|z| > k$, there exists a factorization $z = uvwxy$, with $0 < |vx| \leq |vwx| < k$, such that $uv^n wx^n y \in L$ for every $n \geq 0$.*

Proof: Call a parse tree "loopless" if there is no nonterminal X such that some node labeled X strictly dominates another node labeled X. For any fixed context-free grammar, there exist only finitely many loopless parse trees. (This can be seen by induction on the number r of nonterminals in the grammar: there are only finitely many choices for the root nonterminal X, and finitely many choices for the rule that gives its immediate descendants; each of these descendants then dominates a loopless parse tree that contains no X's and so is generated by a context-free grammar with $r - 1$ nonterminals, and we can apply the induction hypothesis.) Hence, if we choose l large enough, then any derivation of a string z with $|z| > l$, from any nonterminal, has no loopless parse trees. Let m be the largest number of terms on the right-hand side of any production, and let $k = lm$.

For any z with $|z| > k$, let T be an S-parse tree with as few vertices as possible. By the above, there is some nonterminal X such that one X node of T dominates another X node. Choose some two such nodes such that the upper one is as low as possible. Let the terminal string dominated by the lower X be w; then the terminal string dominated by the upper X

is vwx and the terminal string dominated by the root S is $uvwxy$, for some $u, v, x, y \in \Sigma^*$. It follows that $S \overset{*}{\Rightarrow} uXy$, $X \overset{*}{\Rightarrow} vXx$, and $X \overset{*}{\Rightarrow} w$, so we have

$$S \overset{*}{\Rightarrow} uXy \overset{*}{\Rightarrow} uvXxy \overset{*}{\Rightarrow} uvvXxxy \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} uv^nXx^ny \overset{*}{\Rightarrow} uv^nwx^ny$$

for any $n \geq 0$. Moreover, at least one of v, x is nonempty, since otherwise we could replace the subtree dominated by the upper X with the subtree dominated by the lower X, thus creating a parse tree of z with fewer nodes than T, a contradiction. And by our choice of the X nodes, each of the immediate descendants of the upper X dominates a loopless subtree; each such subtree yields a substring of length $\leq l$, and there are at most m of these immediate descendants, so that the string vwx dominated by the upper X has length $\leq lm = k$, as required. □

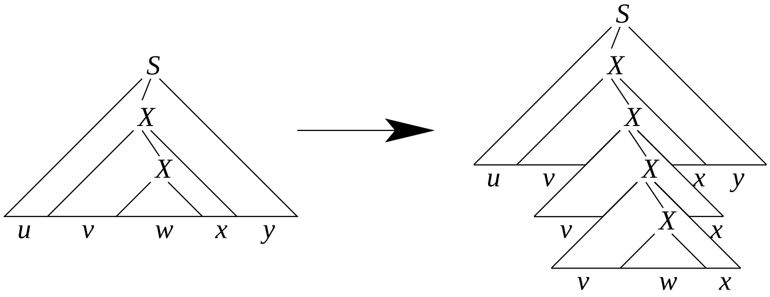


Figure 1: The Pumping Lemma

This basic result can be used to show, for example, that although the language $\{\mathrm{a}^n\mathrm{b}^n\}$ is context-free, the language $\{\mathrm{a}^n\mathrm{b}^n\mathrm{c}^n\}$ is not: as with the proof of the non-regularity of $\{\mathrm{a}^n\mathrm{b}^n\}$, we consider the various possible locations of the subwords v, x and check that every case gives a contradiction. The result can also be used to show that the "copy" language $\{ww \mid w \in \Sigma^*\}$ is not context-free, as long as $|\Sigma| \geq 2$ (for example, consider the strings $z = \mathrm{a}^n\mathrm{b}^n\mathrm{a}^n\mathrm{b}^n$, where n is sufficiently large). On the other hand, the language $\{ww^R\}$ is context-free, where w^R denotes the string obtained by reversing w. Indeed, such a language is generated by the productions $S \rightarrow aSa$ (for each $a \in \Sigma$) and $S \rightarrow \epsilon$. One often says that the language $\{ww^R\}$ contains "nested dependencies", whereas the language $\{ww\}$ contains "crossing dependencies." See the figure.

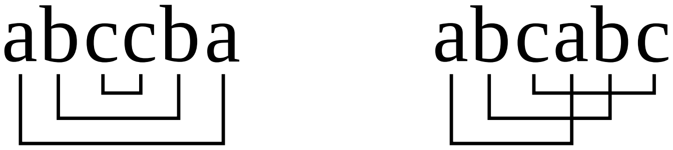


Figure 2: Nested (left) and crossing (right) dependencies

Now, let's move on in our classification of the rewriting systems, because there is more to be done. The next stop is the *context-sensitive grammars*. These are grammars in which every rule is of the form

$$sAt \to swt, \tag{5}$$

where $A \in V$, $s, t \in (V \cup \Sigma)^*$, and $w \in (V \cup \Sigma)^+$. Thus, we can see the motivation for the terms "context-free" and "context-sensitive": a production of a context-sensitive grammar can allow us to rewrite A as the string w, but only in a certain "context" of neighboring symbols; context-free rules contain no such restrictions. These rules were originally motivated by the desire for a concise expression of subcategorization restrictions for lexical items; for example, one might allow the rewrite NP → *John* immediately before an animate-subject verb like *sat*, but not before *happened* ([17]). Admittedly this is a somewhat crude way to express such restrictions, but at least it's a first shot.

Context-sensitive languages are, as one would expect, more general than the context-free languages; for example, $\{\mathrm{a}^n\mathrm{b}^n\mathrm{c}^n \mid n > 0\}$ is not a context-free language, but it is generated by the following context-sensitive rules:

$$S \to \mathrm{a}SBC \qquad S \to \mathrm{a}BC \qquad CB \to DB \qquad DB \to DC \qquad DC \to BC$$

$$\mathrm{a}B \to \mathrm{ab} \qquad \mathrm{b}B \to \mathrm{bb} \qquad \mathrm{b}C \to \mathrm{bc} \qquad \mathrm{c}C \to \mathrm{cc}.$$

Notice that we have allowed only rewriting rules $sAt \to swt$ where $w \neq \epsilon$; in particular, a context-sensitive grammar as defined above can never generate the string ϵ. More conventionally, context-sensitive grammars are allowed to contain the rule $S \to \epsilon$, but only on the condition that S never appears on the right side of any production. If instead we allow rules of the form $sAt \to swt$ for any $s, w, t \in (V \cup \Sigma)^*$, the resulting grammars are called *context-sensitive with erasing*. It turns out that there exists a context-sensitive-with-erasing grammar for any recursively enumerable language (i.e. any L for which there exists a Turing machine that will successively print all the strings of L in some order). In fact, it is not too difficult, given a Turing machine, to construct a context-sensitive-with-erasing grammar that simulates the machine's operation. Conversely, any context-sensitive-with-erasing language must be recursively enumerable — in fact, any rewriting system at all determines a recursively enumerable language, since one can simply enumerate all possible finite sequences of rule applications.

In contrast, we can see that context-sensitive grammars can determine at most decidable languages (i.e. languages L for which there exists a Turing machine that determines, in a finite amount of time, whether or not a given string w is in L). Indeed, the rules of a context-sensitive grammar cannot shorten a string (except for the trivial $S \to \epsilon$), so for any string, there are at most finitely many sequences of rule applications that need to be checked. One can also show, by a Cantor-style diagonalization argument, that there are decidable languages that are not context-sensitive.

The classes of languages we have described make up what is often referred to as the *Chomsky hierarchy*, shown in Figure 3. This hierarchy is widely used as a measurement of the complexity of formal languages. (Sometimes further intermediate levels, not discussed here, are also included, see [33, ch. 1].)

Now that we have defined all of these language classes, we should give some indications of why they are interesting. We have seen that there are linguistic motivations for the forms of, say, context-free and context-sensitive rules. But why should a mathematician be interested

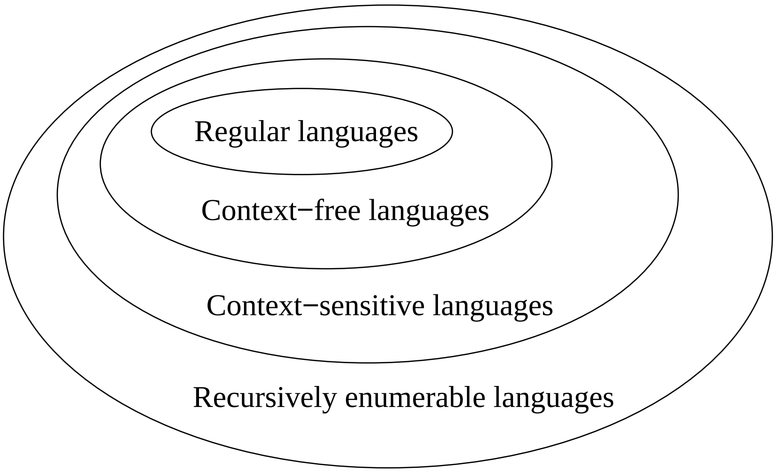


Figure 3: The Chomsky hierarchy

in these classes of languages, as opposed to others that someone might come up with? One reason is that each class has a variety of closure properties that make it a natural class of languages to look at. For example, given a fixed alphabet Σ, and languages L_1, L_2 over Σ, one can define the following operations:

- *union*: $L_1 \cup L_2$,
- *concatenation*: $L_1 L_2 = \{vw \mid v \in L_1, w \in L_2\}$,
- *Kleene star*: $L_1^* = \{w_1 \cdots w_n \mid n \geq 0;\ w_1, \ldots, w_n \in L_1\}$ (i.e. the smallest submonoid of Σ^* containing L_1; note that this is distinct from the free monoid generated by L_1).

Then we have

Theorem 3 *(Kleene's Theorem) The class of (nonempty) regular languages over Σ is the smallest class of languages containing $\{a\}$, for each $a \in \Sigma$, and closed under union, concatenation, and Kleene star.*

The regular languages are also closed under complementation (in Σ^*). We will not give the proofs of these statements here, as we have other things to do right now.

Having thus provided a reason to like the class of regular languages, we can define some more desirable properties of a class of languages. Such a class $\mathcal{F}$ is a *full Abstract Family of Languages (AFL)* [59] if it contains some nonempty language, and is closed under concatenation and union, as well as the following operations:

- *homomorphism*: if L is a language over Σ, and $\phi : \Sigma^* \to \Sigma'^*$ is a homomorphism of monoids (for some alphabet Σ'), then the language $\phi(L)$ over Σ' is in $\mathcal{F}$ whenever L is;
- *inverse homomorphism*: if $\phi : \Sigma'^* \to \Sigma^*$ is a homomorphism of monoids, then $\phi^{-1}(L)$ (a language over Σ') is in $\mathcal{F}$ whenever L is;

- *regular intersection*: if $L \subseteq \Sigma^*$ is in $\mathcal{F}$ and R is any regular language over Σ, then $L \cap R \in \mathcal{F}$.

Note that we do not require an AFL to be closed under intersection or complementation.

The upshot is that the regular languages, context-free languages, and recursively enumerable languages are all full AFLs. In fact, one can show ([59, ch. IV]) that these classes are all *principal* full AFLs; a full AFL $\mathcal{F}$ is principal if there exists $L \in \mathcal{F}$ such that $\mathcal{F}$ is the smallest full AFL containing L. The class of context-sensitive languages is not a full AFL, but if we weaken the homomorphism condition to closure under homomorphisms satisfying $\phi^{-1}(\epsilon) = \{\epsilon\}$, then the conditions are met (such a family of languages is often simply called an *AFL*). Although some of the AFL conditions are easy to see (for example, that each class of languages is closed under unions), the full proof that these classes have all of the AFL properties is fairly substantial, and it would be a digression for us to present it here. However, we will make crucial use of some of the AFL properties in the subsequent application of formal language theory to natural language, so we refer the reader to [33] or [59] for the proofs.

The development of formal language theory has been tightly bound up with the theory of automata, and so a second reason that the classes of the Chomsky hierarchy are of mathematical interest is that each is associated with a standard class of formal automata. Any reader acquainted with these automata is probably already sufficiently familiar with this whole theory not to be reading this section, but for the sake of completeness we will at least outline the correspondence. A language L (over a fixed alphabet) is regular if and only if there is some *finite-state automaton* that accepts the strings of L and no others. A finite-state automaton may be envisioned as a machine that moves in one direction along a read-only tape, possibly changing states at each step in a manner determined by its previous state and the symbol it reads on the tape. Some of the states are designated as "accepting states," and the machine accepts a string precisely if it is in an accepting state just after passing the end of the string. The context-free languages are precisely the sets of strings accepted by *pushdown automata*, automata that can move in one direction along a read-only tape and are equipped with a memory stack, so that symbols may be pushed onto or read off of the top of the stack. Recursively enumerable languages, of course, correspond to Turing machines, and context-sensitive languages correspond to Turing machines that operate in linear space (with respect to the size of the input string).

Given this connection with the theory of computation, it is not surprising that the classes pose parsing problems of different degrees of complexity. It is relatively easy to show that, for any regular language L, there exists a decision machine for L (i.e. a Turing machine that determines whether or not its input is in L) that operates in linear time. If L is context-free, then there exists a decision machine — in fact, a parser (a machine that computes, in suitable form, an S-parse tree) — operating in polynomial time. If L is context-sensitive, a decision machine operating in exponential time exists. For arbitrary recursively enumerable languages, a decision machine does not necessarily exist at all.

Now that we have given some indication as to why the classes of the Chomsky hierarchy are standard landmarks for measuring the complexity of formal languages, it is time to see how to apply this business to human languages. We will try to see where on this scale the class $\mathcal{L}^*$ of human languages lies; this will enable us to assess a theory of language by determining where the $\mathcal{L}$ it generates lies, and seeing how well the two match up. Of course we do not ultimately expect $\mathcal{L}^*$ to coincide with any level of the Chomsky hierarchy, or even with any class of formal languages that can be given a reasonably succinct description, since Universal Grammar is a rich and complex device. However, we can at least attempt to find upper and lower bounds — the smallest "reasonable" class of formal languages that contains $\mathcal{L}^*$, and the largest "reasonable" class that is disjoint from it. We will thus give a brief survey of the arguments that have been advanced toward this goal. Our exposition here is loosely modeled on that of [64].

It is safe to say that natural languages are not regular. As we have seen from the Iteration Theorem, regular languages cannot capture unbounded numbers of nested dependencies, which occur in every natural language. For example, consider the following sentences (embedded sentences are bracketed for clarity):

the cat is gray
the cat that [the dog chased] is gray
the cat that [the dog that [the platypus heard] chased] is gray
⋮

These sentences are all grammatical, even if longer ones rapidly lead to performance judgments of unacceptability. Sentences of this form require that the number of nouns equal the number of verbs, but allow this number to be unbounded; this is a constraint that a regular language cannot satisfy. Using the closure properties of regular languages, one can even prove (granting the data) that English is not regular: if we intersect English with the regular language

the cat { *that the dog, that the platypus* }* { *heard, chased* }* *is gray*

and then apply the homomomorphism sending each noun to a, each verb to b, and everything else to ϵ, the resulting language is $\{\mathrm{a}^n\mathrm{b}^n \mid n \geq 1\}$. As we have seen, this language is not regular, so English cannot be regular either. (For some entertaining examples of nested dependencies in natural language, see [53, ch. 4].)

The question of whether natural languages are context-free is more interesting. Context-free grammars seem to be a natural model for much of human language, as illustrated by our earlier example (4). However, it seems equally clear that plenty of linguistic phenomena are not readily accounted for by context-free rules — namely, the phenomena that are instead described using movement in mainstream generative grammar. For example, it is not apparent how to account for the position of (say) the object *what* in *What do you think is going to happen next?* or *What do you think she said he was hoping for you to buy?* using a context-free grammar for English. However, it also is not at all immediate that such phenomena prevent the language from being context-free. A number of fallacious arguments were advanced early on to claim that natural language was not context-free; Pullum and Gazdar, in their classic paper [55], debunked all of the diverse arguments that had been made until that time.

The standard argument now used against natural language being context-free uses data from Dutch and is originally credited to Huybregts (see [37], [11]). Embedded clauses in Dutch have subject-object-verb word order:

Jan	*zegt*	*dat*	*Piet*	*Marie*	*zag*
Jan	says	that	Piet	Marie	saw

‘Jan says that Piet saw Marie’

The relevant finding is that a verb that takes both an NP and VP as complement comes before, not after, its VP complement:

Jan	*zegt*	*dat*	*Piet*	*Marie*	*zag*	*zwemmen*
Jan	says	that	Piet	Marie	saw	swim

‘Jan says that Piet saw Marie swim’

Moreover, this construction can be applied recursively:

. . . dat	*Piet*	*de leraar*	*Marie*	*zag*	*helpen*	*zwemmen*
. . . that	Pier	the teacher	Marie	saw	help	swim

‘. . . that Piet saw the teacher help Marie swim’

Evidently, then, we obtain strings consisting of n noun phrases followed by n verbs, for arbitrarily large n, such that the ith NP is the subject of the ith verb. Thus we have unbounded crossing dependencies, which is impossible for a context-free language.

As noted by Pullum and Gazdar, this argument is not really quite adequate: because the verbs (except for the first) are infinitives and so have no agreement restrictions, there are no real crossing dependencies at the string level — they exist at the level of parse trees, but we are not considering these to be part of the language. We thus get a language that is basically of the form $\{\mathrm{a}^n\mathrm{b}^n\}$ (ignoring the *Jan zegt dat* at the beginning), which is context-free. However, Shieber [62] patched this up with an analogous argument from the Swiss dialect of German. Swiss German embedded clauses have the same word order as in Dutch, but in addition, Swiss German, like standard German, has case-marking on nouns. Verbs may select whether their objects are accusative case or dative case. Thus, Shieber gives examples such as

Jan	*säit*	*das*	*mer*	*em Hans*	*es huus*	*lönd*	*aastriiche*
Jan	says	that	we	Hans-DAT	the house-ACC	help	paint

‘Jan says that we help Hans paint the house’

Now we have genuine syntactic dependencies; the case of the $(i+1)$th noun is determined by the ith verb. This enables us to conclude that Swiss German is not context-free. A formal argument can be given using regular intersection and homomorphisms to reduce to the copy language $\{ww\}$, analogous to the argument given for regular languages above.

Other arguments for the non-context-freeness of natural language have been made; for example, Culy ([23]) gave a similar copy-language-reduction argument for the morphology of Bambara, a Mande language of Mali. On the basis of these arguments, it is now generally believed that human languages are not context-free (or at least not all of them are). In general, however, any persuasive evidence that has been mustered against the context-freeness of natural language has come from relatively small pockets of specific languages, and has involved copy languages, but not other, more complex violations of context-freeness. This fact (as well as other considerations such as parsing efficiency) has led computational linguists to believe that one need not look “too far” beyond context-free languages to find a suitable upper bound for the complexity of languages in $\mathcal{L}^*$, and in particular that there should be some reasonable class of languages containing $\mathcal{L}^*$ but properly contained in the context-sensitive languages.

In 1985, Joshi ([40]) loosely described three desirable properties for any language that is purported to be in $\mathcal{L}^*$ (or some approximation thereto). Such a language L should

- have limited crossing dependencies — it is not clear exactly what this means, but Joshi gives the example of a grammar system that can generate the language $\{ww\}$ but not the language $\{www\}$;
- satisfy the constant growth property: that is, there is some constant k such that for any $w \in L$, there exists $v \in L$ with $|w| < |v| < |w| + k$;
- be parsable in polynomial time (the definition here is again open to interpretation).

Languages satisfying these conditions are (again, loosely) termed *mildly context-sensitive languages*.

So the hunt is on for grammar systems that give a suitable level complexity to describe natural language. One current prominent candidate is *linear indexed grammars* ([29], a special case of indexed grammars, [2]). These are an extension of context-free grammars wherein each nonterminal is equipped with a form of memory called a *stack*, a finite list of symbols (distinct from the terminals and nonterminals); rules can add to or delete from the end of the stack, and can also apply conditionally depending what symbol is at the end of the stack.

Since we will be using LIG's later, we had better give a full definition. Let V be a finite set of nonterminals, and let I be a finite set of *stack symbols*. Let $W = V \times I^*$; a pair $(X, \eta) \in W$ is notated $X[\eta]$, thought of as a nonterminal X with the stack η attached to it. We can then have a finite set P of productions, all of which are of the forms

$$X[\circ\circ\eta] \to sX'[\circ\circ\eta']t \qquad (X, X' \in V; \quad \eta, \eta' \in I^*; \quad s, t \in (W \cup \Sigma)^*) \tag{6}$$

$$X[\eta] \to w \qquad (X \in V; \quad \eta \in I^*; \quad w \in \Sigma^*). \tag{7}$$

(The $\circ\circ$ in (6) is to be thought of as just a notational device, just like the $\to$.) Now we say that $x \overset{*}{\Rightarrow} y$ iff there exists a sequence $x = x_0, x_1, \ldots, x_n = y$ such that for each i, either

$$x_i = rX[\zeta\eta]u, \;\; x_{i+1} = rsX'[\zeta\eta']tu$$

$$(\text{some } r, u \in (W \cup \Sigma)^*, \zeta \in I^*, \text{ and } X[\circ\circ\eta] \to sX'[\circ\circ\eta']t \text{ in } P)$$

or

$$x_i = rX[\eta]u, \;\; x_{i+1} = rwu \qquad (r, u \in (W \cup \Sigma)^*, \text{ and } X[\eta] \to w \text{ in } P).$$

Thus, rules of type (6) rewrite X as the string $sX't$, conditional on the last few symbols in X's stack, and pass on the rest of the stack to the designated successor X'. Rules of the type (7) simply rewrite X as a particular terminal string, conditional on the exact value of X's stack. Finally, if $S \in V$ is a designated start symbol, then the quadruple (V, I, S, P) is a *linear indexed grammar*, and the language it generates is $\{w \in \Sigma^* \mid S[\,] \overset{*}{\Rightarrow} w\}$.

An example, from [29], will help to elucidate why this kind of grammar system is relevant to human languages. The following indexed grammar can be seen to generate the copy language $\{ww\}$ over the alphabet $\{\mathrm{a}, \mathrm{b}\}$, and thus captures the kind of crossing dependencies we see in Dutch and Swiss German. There are two stack symbols, call them α, β, and nonterminals S, T. The rules are

$$S[\circ\circ] \to \mathrm{a}S[\circ\circ\alpha] \qquad S[\circ\circ] \to \mathrm{b}S[\circ\circ\beta] \qquad S[\circ\circ] \to T[\circ\circ]$$

$$T[\circ\circ\alpha] \to T[\circ\circ]\mathrm{a} \qquad T[\circ\circ\beta] \to T[\circ\circ]\mathrm{b} \qquad T[\,] \to \epsilon.$$

After some doubt over their relevance to natural language ([29]), linear indexed grammars came into the limelight in 1988. The crucial event was when Vijay-Shanker and Weir showed that they generated the same class of languages as several other grammar systems, which had been developed independently by various linguists and computer scientists ([74], [73]; see also [41] for a less precise but more readable exposition). Each of these grammar systems generates a proper superset of the context-free languages, but also meets the three mildly context-sensitive conditions given by Joshi; they all generate the language $\{ww\}$ over any finite alphabet, but not $\{www\}$ — indeed, they can be shown to satisfy a version of the pumping lemma in which the factorization has nine terms rather than five, and the four even-positioned factors are pumped ([71]; see also [48] for a further generalization).

We will briefly mention the other grammar systems studied by Vijay-Shanker and Weir. One of these other systems is head grammars ([57]), which are similar to context-free grammars except that each string (of terminals or nonterminals) is equipped with a division point between two symbols. In addition to the usual context-free rewrite rules, which concatenate substrings and inherit the division point from one of the substrings, we also allow *wrapping* rules, which take two strings and insert the first into the second at the location specified by the division point of the second string. Another system is tree-adjoining grammars ([39], [71]), which give a finite set of initial (ordered, labeled) trees and auxiliary trees, and generate new trees by splicing the auxiliary trees into initial trees; strings of the language are then produced by reading off the leaves of these trees. The last grammar system is (a particular version of) combinatory categorial grammars, which we shall describe in detail soon enough since it will be our focus.

We shall call the languages generated by these systems *linear indexed languages*, for lack of a more neutral term in the literature. The fact that four independently devised (and linguistically motivated) grammar systems all generate the class of linear indexed languages, and that this family properly contains the context-free languages and satisfies Joshi's criteria, strongly suggests that this family is a reasonable approximation to at least an upper bound for $\mathcal{L}^*$. Weir constructed, in [76], an infinite hierarchy (ordered by inclusion) of full AFL's, all properly contained in the context-sensitive languages and containing the context-free languages. The lowest level of Weir's hierarchy consists of precisely the linear indexed languages.

As a result, it is widely believed that the linear indexed languages form the best known upper bound for $\mathcal{L}^*$ in terms of complexity. In keeping with this school of thought, we will consider it to be a desirable trait of a grammar system, for purposes of modeling natural language, that it should generate at most linear indexed languages.

It should be pointed out that there are some who believe that natural languages can fall outside of the linear indexed class. Radzinski ([56]), among others, makes an argument in this direction using names of cardinal numbers, and Kac ([42]) gives an argument based on the use of the word *respectively* in English that could be used to assert that English does not lie in any of the classes of the Weir hierarchy. However, in this author's view, both of these arguments depend on the ungrammaticality of sentences that are not *syntactically* ill-formed, but rather are judged unacceptable for reasons of arithmetic, semantics, or pragmatics. It also seems, particularly in the case of Kac's argument, that the empirical data are so questionable as to lie in the realm where idealizations about language break down, and natural-language acceptability judgments should no longer be applied to the formal languages used as models. Therefore, we will disregard these arguments and continue to treat the linear indexed languages as an appropriate estimate for the complexity of $\mathcal{L}^*$.

A very important point is that our discussion of the complexity of natural language has focused on the *weak* generative capacity of various grammar systems — we treat a language only as a set of strings. One can also compare some grammar systems in terms of *strong* generative capacity, where one treats a language as a set of parse trees, so that in order to be equivalent two grammar systems must not only generate the same sets of strings but also generate the same sets of trees for those strings (under a suitable definition of "sameness"). It is often argued (e.g. [16], [29]) that strong capacity is ultimately more important in linguistics, as generative grammarians strive to accurately describe the structures assigned to natural-language sentences. However, we will focus mainly on weak generation here (although structures will play an important role later, in Section 6). We do this not only because weak generation results are more accessible but also because we do not currently want to presuppose that we know the "correct" tree structure(s) for any given sentence; string acceptability judgments provide much more robust data. We mention strong generative capacity now only to point out that, in principle, the formal tools we use are not necessarily the most powerful ones available for evaluating a linguistic theory.

3 Categorial grammars

In this section, we describe the framework of categorial grammars (CG). We begin by outlining the major differences between CG and more mainstream theories of syntax, in order to place it in a proper linguistic context, and then we present a formal statement of the theory. Categorial grammar traces its origins to the work of the logician Ajdukiewicz ([3]) and of Bar-Hillel ([8], [9]), and it has since evolved into a serious approach to syntax on even footing with standard transformational theories. We give a very brief exposition; see [38], [68], [77] for more extensive treatments of the theory itself as well as the ideas and methodology behind it.

The most distinctive property of categorial grammar that separates it from most of the generative grammar tradition (e.g. the GB program [20], Minimalism [22]) is that it is a non-transformational theory. CG eschews the notion of a fixed number of different levels of representation for sentences. Instead, the grammar puts together successively larger constituents with rules akin to the context-free rules we have seen above, and when it is finished, the resulting string is a sentence. In particular, the grammar does not explicitly operate on trees, unlike essentially all transformational grammars that have been proposed; it only assembles strings. We may use parse trees as a notational device to describe the process by which strings are put together (again, as in the example of context-free grammars in the previous section), but they are not data structures available to the grammar.

Expositions of CG also emphasize the production of sentences as a dynamic process, wherein the constituents are successively put together (although this is a property of the grammar, not necessarily of the computation). A distinct benefit of the theory is that the syntax is accompanied by a model-theoretic compositional semantics, and semantic rules apply in tandem with the syntactic rules, so that each constituent is naturally assigned a meaning at the time it gets assembled. However, this will not concern us here, since our focus is exclusively on syntax.

Prohibiting transformations seems at first glance to be extremely restrictive, but as we shall see, CG provides means of accounting for all sorts of linguistic phenomena that are commonly explained via movement. For example, passive sentences are not derived by the usual method of first constructing their active counterparts and then applying transformations. Instead, passivization occurs at the lexical level; thus a verb such as English *smush* gives rise to a verb *be smushed*, with concomitant changes to the syntactic subcategorization, the semantics (rearranging argument structure), and the morphology (requiring *by*-case-marking on the agent). The similarity in meaning between *Lenny smushed the banana* and *The banana was smushed by Lenny* then arises from the semantic relationship between *smushed* and *was smushed*. (This idea was originally proposed by Dowty [25] in a different grammatical framework.)

Similarly, CG does not make use of filters applying at specific derivational levels, such as the Case filter, θ-criterion, Extended Projection Principle, and so forth from GB. Sentences that are ruled ungrammatical in GB because they fail to meet these criteria are ungrammatical in CG simply because, at some point in the desired derivation, two constituents cannot to be put together.

Another distinguishing property of gategorial grammar is its heavy reliance on the lexicon as the locus of syntactic information. The basic motivation here is the realization that this use of the lexicon is often inevitable. For example, consider the subcategorization requirements of the verbs *like* and *dislike*. The former can take a noun phrase or an infinitival phrase as complement, but the latter can only take a noun phrase: *I like to tip cows* is grammatical; **I dislike to tip cows* is not. This sort of information does not seem to be predictable on the basis of meanings, which means that it needs to be stored in the lexicon. One can come up with plenty of further examples along these lines.

Categorial grammar is a framework that can express this kind of subcategorization efficiently. Briefly put, the categories assigned to lexical items consist of a few primitive categories, and of compound categories of the form X/Y; an item of category X/Y can combine with an expression of category Y to form an expression of category X. Thus, for example, verbs can select for different kinds of complements, depending on what argument category (i.e. what value of Y) they are assigned. So, if VP were one of our primitive categories, we might represent the *like*-*dislike* distinction by assigning *like* the categories VP/NP and VP/InfP, but giving *dislike* only VP/NP. In fact, as we shall soon see, this idea of creating categories using slashes can be taken much further, and ultimately we have a *fully lexicalized* theory of syntax — one in which *all* variation between languages is contained in the lexicon, with the rules themselves few, elegant, and universal.

Now then, let's give a full statement of the formalism. We begin with a simple form that is sometimes called *classical categorial grammar* (more or less the formulation given in [8]; these systems are also sometimes called *AB systems*, after Ajdukiewicz and Bar-Hillel). Our definitions are modeled on those used by Kanazawa in [43].

Given a finite set C of symbols, called *primitive categories*, we define the *category space over* C to be the set V generated by taking the closure of C under two generic binary operations, notated ${/}_{\mathrm{L}}$ and ${/}_{\mathrm{R}}$. More formally, we may model V as follows: let ${/}_{\mathrm{L}}$ and ${/}_{\mathrm{R}}$ be two symbols; recursively define sets V_i by taking

$$V_0 = C, \qquad V_{i+1} = V_i \cup (V_i \times \{{/}_{\mathrm{L}}, {/}_{\mathrm{R}}\} \times V_i).$$

Then let

$$V = \cup_{i=0}^{\infty} V_i.$$

We then use notation such as $X{/}_{\mathrm{L}}Y$ to represent the triple $(X, {/}_{\mathrm{L}}, Y) \in V$. (Our slash notation comes from [38]; the slashes \ and / are more conventional, but also more confusing, since some authors use $X\backslash Y$ where others write $Y\backslash X$.) Note that these operations are nonassociative — $(X{/}_{\mathrm{R}}Y){/}_{\mathrm{R}}Z$ is different from $X{/}_{\mathrm{R}}(Y{/}_{\mathrm{R}}Z)$ — and certainly noncommutative. Elements of V will be called *categories*, or *types*.

For a given terminal alphabet Σ, suppose that f is any function from Σ to the set of finite subsets of V; we will call it the *assignment function*. Also suppose that S is a distinguished element of C (the *start symbol*). Then the triple (C, S, f) is a *classical categorial grammar* over Σ. The language it generates is defined as follows: imagine that we have an infinite set of context-free rules,

$$X \to X{/}_{\mathrm{R}}Y\ Y, \qquad X \to Y\ X{/}_{\mathrm{L}}Y \tag{8}$$

for all $X, Y \in V$, and also the rule

$$X \to a \tag{9}$$

whenever $a \in \Sigma$ and $X \in f(a)$. Then the language generated by the classical categorial grammar (C, S, f) is just $\{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$. (In the definition of rewriting systems in Section 2, we required the set of rules to be finite, but clearly the definition extends to infinite sets of rules.) We should mention that in the categorial grammar literature the convention is to notate these rule schemata in the reverse direction, thus writing $X{/}_{\mathrm{R}}Y\ Y \to X$; this practice will be ignored here, in favor of consistency with the conventions for other types of rewrite systems.

For an example, here is a classical categorial grammar for a small fragment of English. Let $C = \{\mathrm{S}, \mathrm{NP}, \mathrm{N}\}$. (In linguistic examples, the start symbol S will always be the sentence category S.) Let $\Sigma = \{\mathit{John}, \mathit{mouse}, a, \mathit{the}, \mathit{slept}, \mathit{poked}\}$. Let f be defined as follows:

$$f(\mathit{John}) = \{\mathrm{NP}\} \qquad f(\mathit{mouse}) = \{\mathrm{N}\} \qquad f(a) = \{\mathrm{NP}{/}_{\mathrm{R}}\mathrm{N}\} \qquad f(\mathit{the}) = \{\mathrm{NP}{/}_{\mathrm{R}}\mathrm{N}\}$$

$$f(\mathit{slept}) = \{\mathrm{S}{/}_{\mathrm{L}}\mathrm{NP}\} \qquad f(\mathit{poked}) = \{(\mathrm{S}{/}_{\mathrm{L}}\mathrm{NP}){/}_{\mathrm{R}}\mathrm{NP}\}$$

Then we have derivations such as the following (where, for succinctness, we have skipped over the terminal substitution rules):

$$\begin{aligned} \mathrm{S} \Rightarrow \mathrm{NP}\ \mathrm{S}{/}_{\mathrm{L}}\mathrm{NP} &\Rightarrow \mathrm{NP}{/}_{\mathrm{R}}\mathrm{N}\ \mathrm{N}\ \mathrm{S}{/}_{\mathrm{L}}\mathrm{NP} \\ &\overset{*}{\Rightarrow} \mathit{The\ mouse\ slept} \end{aligned}$$

$$\begin{aligned}\mathrm{S} \Rightarrow \mathrm{NP}\ \mathrm{S}/_{\mathrm{L}}\mathrm{NP} \Rightarrow \mathrm{NP}\ (\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP}\ \mathrm{NP} &\Rightarrow \mathrm{NP}\ (\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP}\ (\mathrm{NP}/_{\mathrm{R}}\mathrm{N})\ \mathrm{N} \\ &\overset{*}{\Rightarrow} \textit{John poked a mouse}\end{aligned}$$

and so forth.

As in our earlier context-free example, it is easy to obtain recursion in categorial grammars. For example, we can introduce a terminal *thinks* and set $f(\textit{thinks}) = \{(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{S}\}$. Then using the derivation

$$\mathrm{S} \Rightarrow \mathrm{NP}\ \mathrm{S}/_{\mathrm{L}}\mathrm{NP} \Rightarrow \mathrm{NP}\ (\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{S}\ \mathrm{S}$$

we can embed sentences in other sentences. We thereby can describe sophisticated cogitations on the part of man and mouse, generating strings such as *John thinks a mouse thinks John poked the mouse*. Notice also that so far we have only assigned one category to each terminal via f. This is not obligatory. For example, by setting

$$f(\textit{thinks}) = \{(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{S}, \mathrm{S}/_{\mathrm{L}}\mathrm{NP}\}$$

we can include both transitive and intransitive *thinks* in our lexicon. (In actuality, though, we may want to think of these as distinct lexical items — thus as distinct terminals — that happen to be phonologically identical.)

If w is a string and X a category, we say that *w is an X* (or *w is of category X*) to mean simply that $X \overset{*}{\Rightarrow} w$. This expresses the intuition that (for example) *the mouse* is a noun phrase (NP), just like *John*, even though one is specified that way directly in the lexicon (i.e. by the assignment function f) and the other is not.

The resemblance between classical CG rule schemata and context-free rules — and, perhaps, between the derivations of strings in our CG example above and the derivations of strings in our earlier context-free example — suggests that the two systems should generate similar languages. In fact, as we shall see (Theorem 4), the class of languages generated by classical categorial grammars consists precisely of the context-free languages not containing ϵ. This quite simple result is a good sign for the formal tractability of the classical CG framework, but not a good sign for its linguistic adequacy.

Accordingly, numerous further rules have been proposed, generally motivated by specific natural-language phenomena; these rules, we will later see, enable the generation of non-context-free languages. The broad term *combinatory categorial grammars* (CCGs) is used to describe systems similar to classical CGs but also encompassing some or all of these rules. The examples we will discuss come largely from [69]. We have included examples of constructions from natural languages, mostly English, to indicate why these rules have been proposed; if we were to bring them blindly out of thin air, it might be unclear why we introduced these rules and not others, and hopefully the examples will provide some motivation. In our examples, we will not spell out explicitly the category of every word; we will expect these to be clear from context, e.g. count nouns are N; determiners such as *the* are $\mathrm{NP}/_{\mathrm{R}}\mathrm{N}$; intransitive verbs are $\mathrm{S}/_{\mathrm{L}}\mathrm{NP}$; transitive verbs are $(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP}$, $(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{S}$, and so forth according to their type of complement. We will also somewhat informally talk about "applying" a rule of the form, say, $X \to Y\ Z$ in order to obtain an expression of category X from two expressions of categories Y, Z; as previously noted, the arrow is merely a notational device, and we sometimes find it easier to conceptualize the derivation of a sentence via steps going from the right side of a rule to the left side, rather than vice versa.

The basic rule schemata we have already given in (8) are generally known as *function application* rules (the origin of the name has to do with the corresponding semantics). We next introduce two types of rules: we have the *function composition* rules

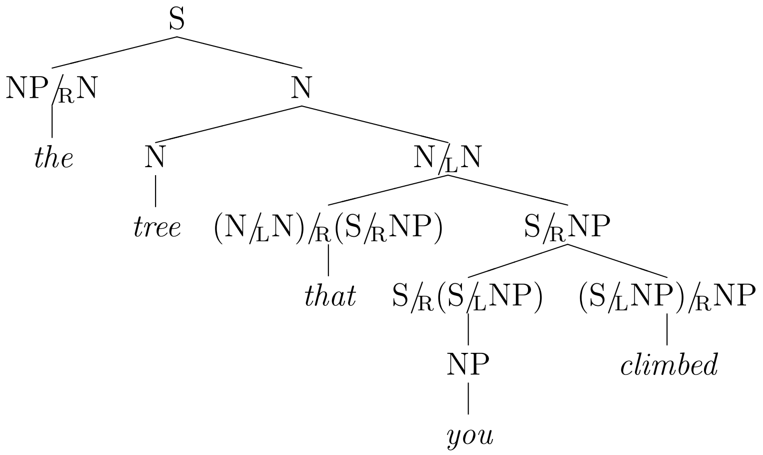
$$X/_{\mathrm{R}}Z \to X/_{\mathrm{R}}Y\ Y/_{\mathrm{R}}Z \qquad X/_{\mathrm{L}}Z \to Y/_{\mathrm{L}}Z\ X/_{\mathrm{L}}Y, \tag{10}$$

for all categories X, Y, Z, and the *type-lifting* rules

$$X/_{\mathrm{R}}(X/_{\mathrm{L}}Y) \to Y \qquad X/_{\mathrm{L}}(X/_{\mathrm{R}}Y) \to Y. \tag{11}$$

In isolation, these rules are not terribly useful. However, when used in combination, they can account for a wide range of what are commonly called "extraction" phenomena. For example, we can use them to provide a natural description of relative clauses in English. Let *that* have type $(\mathrm{N}/_{\mathrm{L}}\mathrm{N})/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{R}}\mathrm{NP})$; then we can derive a noun phrase such as *the mountain that you climbed* as

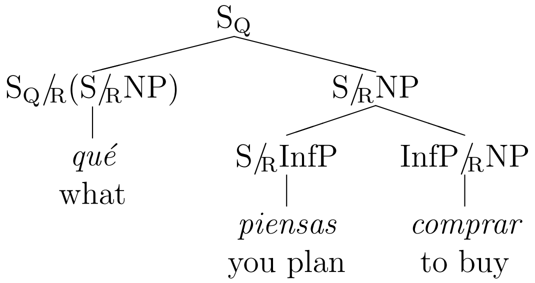
(12)



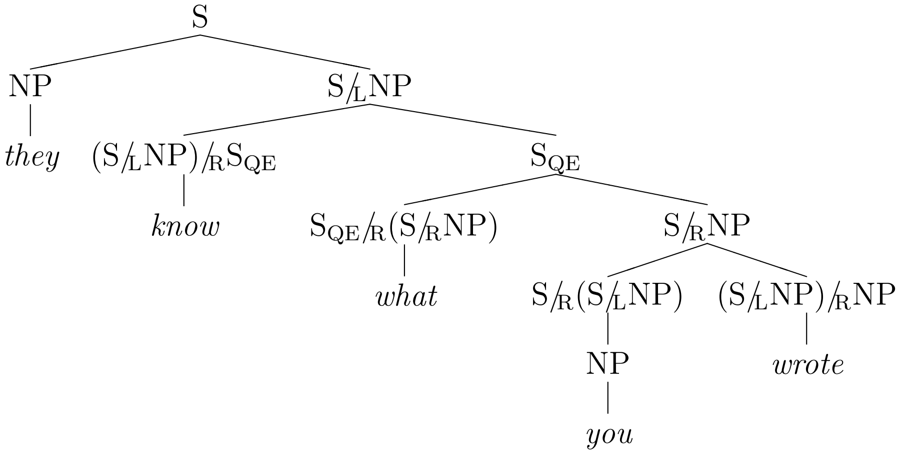
(One could then incorporate this into a longer sentence such as *I saw the mountain that you climbed* by the usual function application rules.) The crux of the derivation is that *you climbed* needs to be of category $\mathrm{S}/_{\mathrm{R}}\mathrm{NP}$; this is impossible using function application alone but can be accomplished by type-lifting *you* and applying composition.

The $\mathrm{S}/_{\mathrm{R}}\mathrm{NP}$ construction can be used to account for other cases of extraction, such as wh-questions. For example, we can derive the Spanish question *¿Qué piensas comprar?* 'What do you plan to buy?' if we give *qué* the category $\mathrm{S_Q}/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{R}}\mathrm{NP})$. (Here $\mathrm{S_Q}$ represents a question, and InfP represents an infinitival verb phrase.)

(13)

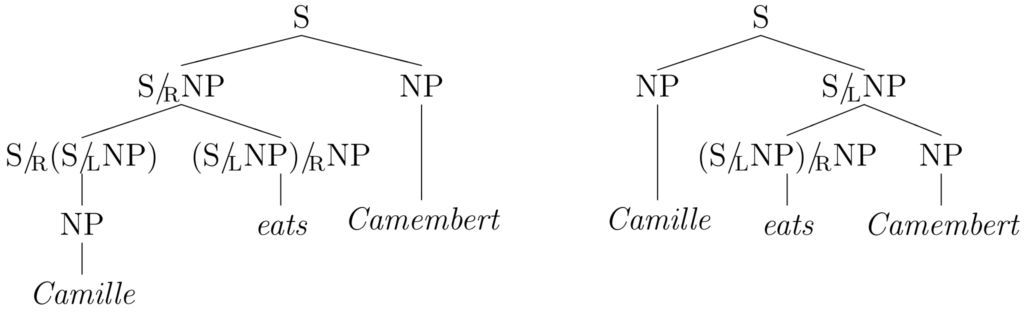


Here we have only needed to employ composition, since there is no overt subject to type-lift. In English direct questions (and in Spanish questions where there is an overt subject), the situation is messier because of subject-auxiliary inversion. However, for now we at least have the apparatus to handle embedded questions in English: if *know* subcategorizes for an $\mathrm{S_{QE}}$ argument (QE for "embedded question") and *what* is of type $\mathrm{S_{QE}}/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{R}}\mathrm{NP})$, then we get sentences like



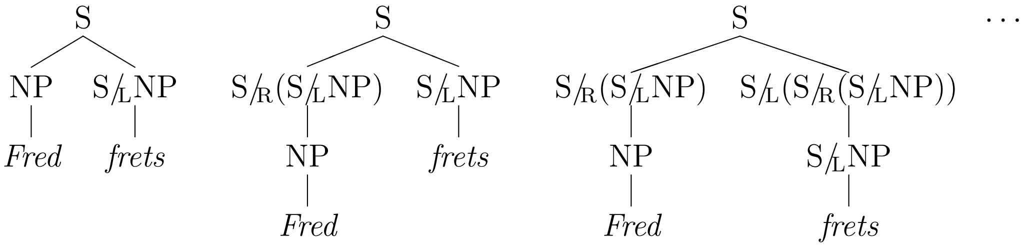
(14)

It is worth pointing out that these new rules also give many new derivations for sentences that were already derivable in classical-land. For example, a parse tree for a simple sentence such as *Camille eats Camembert* can now be "left-branching" or "right-branching":



(15)

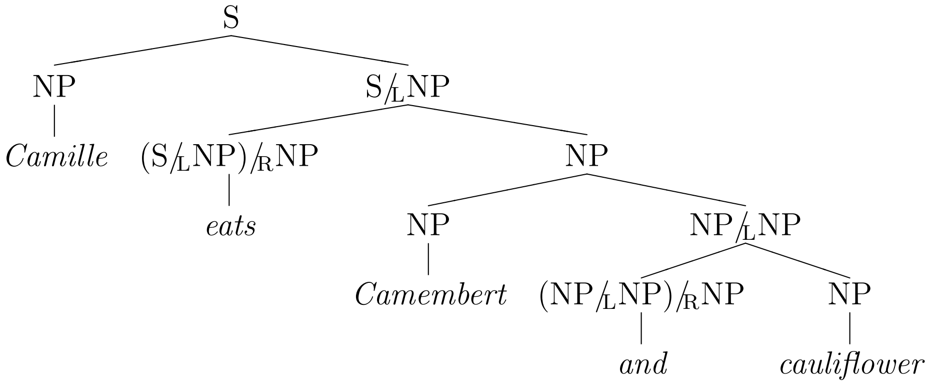
In fact, with an unbounded number of applications of type-lifting, we can construct infinitely many parse trees for a single sentence such as *Fred frets*:



(16)

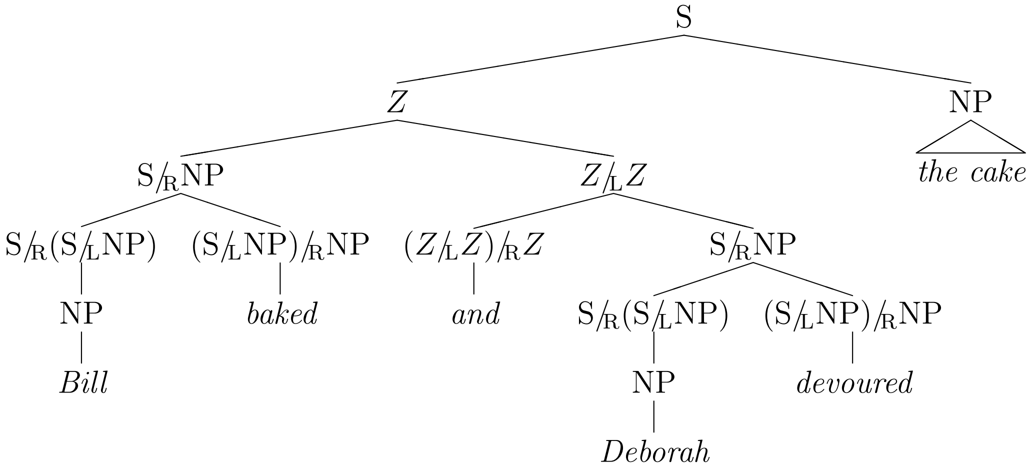
However, this will not concern us, since we are interested only in knowing whether or not a string is derivable, not how many derivations it has. (It might seem distasteful that there are infinitely many possible parses, and indeed it does pose a problem for the construction of parsers. However, it turns out that there is no ambiguity in meaning; the workings of the semantics are such that all the trees for *Fred frets* above give logically equivalent meanings. The standard approach to the parsing problem in this situation involves giving just the most parsimonious parsings. While eating parsley.)

Coordination is usually accounted for by letting *and* and *or* be assigned the categories $(X /_{\mathrm{L}} X) /_{\mathrm{R}} X$, for some large collection of possible values of X (some authors, such as [69], even seem to allow a special rule schema allowing any arbitrary X to be used). Thus we can give Camille a more balanced diet:



(17)

A further benefit of composition and type-lifting rules is that they allow easy descriptions of what are usually known as “right-node raising” constructions ([54]). In most versions of generative grammar, the analysis of a sentence such as *Bill baked and Deborah devoured the cake* is nontrivial; but in a CCG framework it is straightforward, since *Bill baked* and *Deborah devoured* are ordinary constituents. (To make the tree fit on the page, we write Z for $\mathrm{S} /_{\mathrm{R}} \mathrm{NP}$.)



(18)

Composition is also used to handle auxiliaries. For example (ignoring verb tensing for simplicity), English *might* could be regarded as an $(\mathrm{S} /_{\mathrm{L}} \mathrm{NP}) /_{\mathrm{R}} (\mathrm{S} /_{\mathrm{L}} \mathrm{NP})$, combining with a verb phrase such as *buy the car* to give *might buy the car*, again of category $\mathrm{S} /_{\mathrm{L}} \mathrm{NP}$. However, it seems as though there should also be a derivation for *might buy* that treats it as a transitive verb, e.g. so that it can coordinate in a sentence such as *Irwin likes and might buy the* *car.* This is no problem if we have a composition rule; the $(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})$ *might* and $(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP}$ *buy* combine to give $(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP}$ *might buy* immediately.

We would like to generalize this to, say, ditransitive verbs; for example, *give* has assigned category $((\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP}$, and we would like to apply composition to obtain *might give* of this same category (without needing to give *might* a new category). We can do this with an extension of the usual composition rules. Accordingly, *generalized function composition* schemata have been proposed: these give rules of the form

$$(\cdots((X/_{\mathrm{R}}Z_1)/_{\mathrm{R}}Z_2)\cdots/_{\mathrm{R}}Z_n) \;\rightarrow\; X/_{\mathrm{R}}Y\;(\cdots((Y/_{\mathrm{R}}Z_1)/_{\mathrm{R}}Z_2)\cdots/_{\mathrm{R}}Z_n) \tag{19}$$

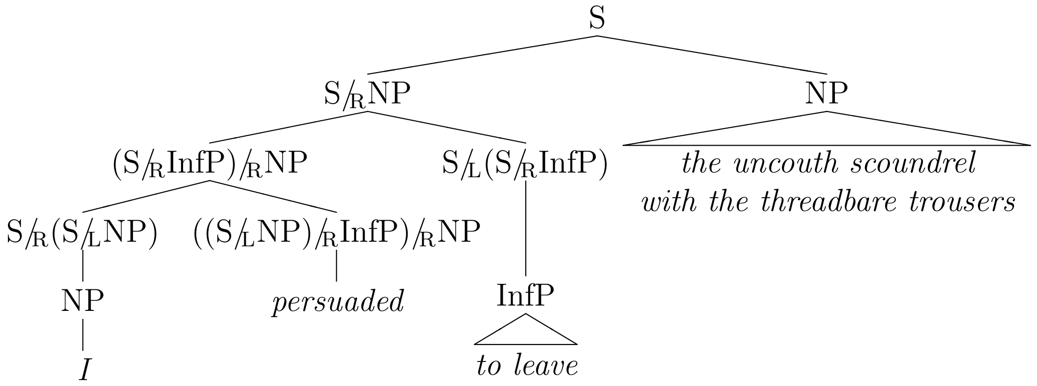
$$(\cdots((X/_{\mathrm{L}}Z_1)/_{\mathrm{L}}Z_2)\cdots/_{\mathrm{L}}Z_n) \;\rightarrow\; (\cdots((Y/_{\mathrm{L}}Z_1)/_{\mathrm{L}}Z_2)\cdots/_{\mathrm{L}}Z_n)\;X/_{\mathrm{L}}Y \tag{20}$$

for arbitrary categories $X, Y, Z_1, \ldots, Z_n \in V$.

The composition rules (19) are more precisely referred to as *harmonic composition* rules. There are also *crossed composition* rules, which combine slash types:

$$X/_{\mathrm{R}}Z \rightarrow Y/_{\mathrm{R}}Z\;X/_{\mathrm{L}}Y \qquad X/_{\mathrm{L}}Z \rightarrow X/_{\mathrm{R}}Y\;Y/_{\mathrm{L}}Z \tag{21}$$

These can be used, for example, to account for the the phenomenon known as "heavy NP shift" in English, if one also allows generalized harmonic composition. The verb *persuaded* is assigned the category $((\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{InfP})/_{\mathrm{R}}\mathrm{NP}$, as in the sentence *I persuaded him to leave.* However, one sometimes finds the *NP* complement rightmost, and such sentences can be derived as follows:



(22)

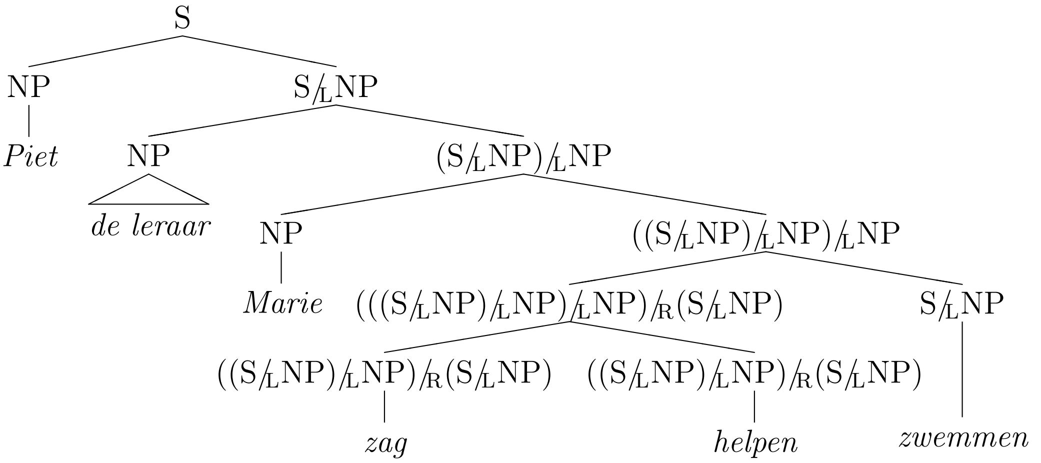
where crossed composition is used to derive the constituent *I persuaded to leave.* (We leave aside the question of how to ensure this happens only when the NP is heavy.)

We can generalize both the harmonic and crossed composition rules to obtain *generalized mixed composition* schemata:

$$(\cdots((X|_1Z_1)|_2Z_2)\cdots|_nZ_n) \;\rightarrow\; X/_{\mathrm{R}}Y\;(\cdots((Y|_1Z_1)|_2Z_2)\cdots|_nZ_n) \tag{23}$$

$$(\cdots((X|_1Z_1)|_2Z_2)\cdots|_nZ_n) \;\rightarrow\; (\cdots((Y|_1Z_1)|_2Z_2)\cdots|_nZ_n)\;X/_{\mathrm{L}}Y \tag{24}$$

for any categories $X, Y, Z_i \in V$ and any $|_1, \ldots, |_n \in \{/_{\mathrm{L}}, /_{\mathrm{R}}\}$. These rules enable us to describe the word order in Dutch embedded clauses (see Section 2):


S
NP
$\mathrm{S}/_{\mathrm{L}}\mathrm{NP}$
Piet
NP
$(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{L}}\mathrm{NP}$
de leraar
NP
$((\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{L}}\mathrm{NP}$
Marie
$(((\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})$
$\mathrm{S}/_{\mathrm{L}}\mathrm{NP}$
$((\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})$
$((\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})$
zag
helpen
zwemmen

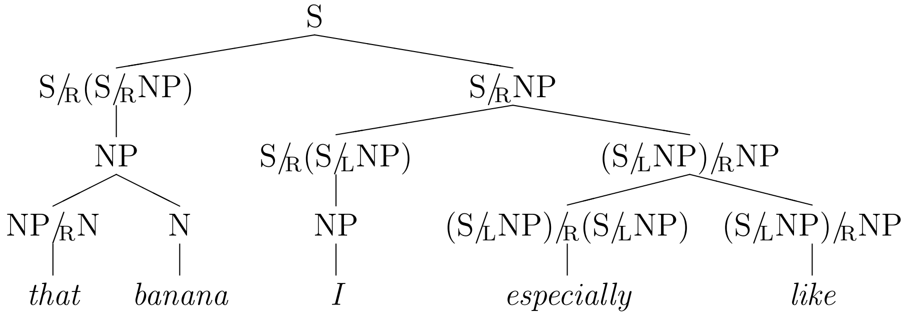
(25)

Here mixed composition is used to obtain the constituent *zag helpen*. (We have omitted the initial *Jan zegt dat*, and have also treated infinitival embedded sentences as S for simplicity.) We can extend this structure to obtain derivations for embedded clauses with arbitrarily many *helpen*-style verbs, using instances of the composition rule only for $n \leq 2$. (The derivation given on [68, p. 141], and reused e.g. by [7, p. 117], requires the use of the composition rule for unbounded n; as we will later see, there are reasons to prefer a structure in which n is bounded.)

Crossed type-lifting rules have also been proposed:

$$X/_{\mathrm{R}}(X/_{\mathrm{R}}Y) \to Y \qquad X/_{\mathrm{L}}(X/_{\mathrm{L}}Y) \to Y \tag{26}$$

In particular, the first of these can describe topicalization, by lifting the topicalized expression ([66]). An example is the English sentence *That banana, I especially like*:


S
$\mathrm{S}/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{R}}\mathrm{NP})$
$\mathrm{S}/_{\mathrm{R}}\mathrm{NP}$
NP
$\mathrm{S}/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})$
$(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP}$
$\mathrm{NP}/_{\mathrm{R}}\mathrm{N}$
N
NP
$(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})$
$(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP}$
that
banana
I
especially
like

(27)

The above are the main rule schemata that have been proposed in CCG frameworks, although there are others (see e.g. [66], [49, ch. 21]). In view of the profusion of rules, one might expect that, in a CCG system using many of them, it would be necessary to somehow restrict the applicability of the rules in order to avoid generating sentences that should be disallowed. Pursuing this direction, the version of CCGs presented in [73] allows the use of rule schemata in which some of the variables are restricted to (finite sets of) specific categories. For example, a CCG system might include the crossed lifting rule $X/_{\mathrm{R}}(X/_{\mathrm{R}}Y) \to Y$ only when Y is one of the types $\mathrm{NP}, \mathrm{InfP}, \mathrm{PP}$. This would prevent the topicalization of arbitrary categories as in **Animal, a llama is an.* A further extension allowed in [73] is to allow rule schemata in which some variables are restricted only to have their *targets* be in certain categories; the target $\tau(X)$ of a category $X \in V$ is defined recursively by

$$\tau(X) = X \ (X \in C); \qquad \tau(X/_{\mathrm{L}}Y) = \tau(X/_{\mathrm{R}}Y) = \tau(X).$$

Thus, we might restrict the possible values of Y in the rule (26) not by specifying a finite list of possible values for Y but by specifying a list of possible values for $\tau(Y)$. Although such a restriction is not uncommon in the categorial grammar literature ([1], [65], [7]), a certain amount of clutter would be necessary in order to accommodate it in our definitions and proofs. We will only remark now that many of our results on generative power can be easily extended to cover rule schemata with target-restricted categories, but the details are omitted for the sake of space and clarity.

A further major concern is with features. Any theory of syntax worth its salt needs to have not only major categories but also features — gender, number, person, tense, and so forth need to be tracked and checked for agreement. For example, the grammaticality difference between *he walks* and **I walks* is expressed using gender features, for which the subject and verb have to agree. The formal theory and computational modeling of feature systems is a field of research in itself ([61], [30]). The basic treatment is as follows: one operates with a fixed set of features (attributes) and a fixed set of possible values for each feature; a *feature structure* then is an assignment of values to some (not necessarily all, and possibly none) of the features. Two feature structures F_1, F_2 can be *unified* to form a new feature structure, whose value for each feature is that given by F_1 or F_2 (if such a value exists); if F_1 and F_2 give inconsistent values for some feature, the unification *fails.*

Linguistic applications of CCGs follow this methodology for handling features. Every appearance of a primitive category is assigned a feature structure, which is written subscripted in brackets. Then, for example, one writes the function application rule as $X_{[F_1]} \to X_{[F_1]}/_{\mathrm{R}}Y_{[F_2]}\ Y_{[F_3]}$, and the rule applies only for feature structures such that F_2 and F_3 unify successfully. Thus certain derivations that would otherwise be grammatical will become impossible on account of feature disagreement. For example, the pronoun *I* would be listed in the lexicon as $\mathrm{NP}_{[1s]}$; this feature structure is shorthand for an assignment of 1 to the feature "person" and *singular* to the feature "number" (we ignore all other features for simplicity). Similarly, *he* would be listed as $\mathrm{NP}_{[3s]}$, while the verb form *walks* would be $\mathrm{S}/_{\mathrm{L}}\mathrm{NP}_{[3s]}$. Thus we have the derivation

$$\mathrm{S} \Rightarrow \mathrm{NP}_{[3s]}\ \mathrm{S}/_{\mathrm{L}}\mathrm{NP}_{[3s]} \overset{*}{\Rightarrow} \textit{he walks}$$

but not

$$\mathrm{S} \Rightarrow \mathrm{NP}_{[1s]}\ \mathrm{S}/_{\mathrm{L}}\mathrm{NP}_{[3s]} \overset{*}{\Rightarrow} \textit{I walks},$$

since the latter derivation is blocked by the inability of $[1s]$ and $[3s]$ to unify.

Again, the subject goes much deeper. For example, the need to handle long-distance agreement phenomena is addressed by categorizing some items as *lexical inheritors*, which propagate features of their arguments. Thus, [38] shows how we may address the gender agreement in sentences like *He tried to kill himself* vs. **She tried to kill himself* by assigning to *tried* the "category" $(\mathrm{S}/_{\mathrm{L}}\mathrm{NP}_\alpha)/_{\mathrm{R}}\mathrm{InfP}_\alpha$. This is really a shorthand for assigning *tried* to two categories, one in which both α's are replaced by $[M]$ and one in which both α's are replaced by $[F]$ (again, ignoring many other features).

Once one integrates features and unification into a CCG framework as described here, the resulting formalism is uncomfortably elaborate for purposes of proving theorems. However, for our formal purposes, there is at least a brute-force way of accommodating features without adding anything to the category apparatus we have already constructed. The idea is as follows: suppose that we have a categorial grammar (C, S, f) (and some fixed set of CCG rule schemata). We would like to rule out certain derivations based on feature agreement. If we have finitely many features, and finitely many possible values for each, then let D be the set of possible feature structures, and let $C' = C \times D$ (which is evidently finite). We then think of an element $(X, F) \in C'$ as the feature-marked category $X_{[F]}$. Our new grammar then assigns each terminal a various categories over C', obtained by attaching feature specifications to the categories originally in $f(a)$.

Thus, for example, if our features include person, number, and case, *the* would be assigned all of the categories $\mathrm{NP}_{[3s,Nom]}/_{\mathrm{R}}\mathrm{N}_{[3s,Nom]}$, $\mathrm{NP}_{[3s,Acc]}/_{\mathrm{R}}\mathrm{N}_{[3s,Acc]}$, $\mathrm{NP}_{[3p,Nom]}/_{\mathrm{R}}\mathrm{N}_{[3p,Nom]}$, and so forth. This approach shows that, as far as formal language theory is concerned, we need not change our operational definition of categorial grammars in order to accommodate feature agreement.

The reader is probably starting to get tired of all these various extensions to CCG, but we should include one more kind of extension here, because it is one that we will be able to cover with our upcoming formalism. It is widely believed that CCGs should be formulated with more slash operators than just $/_{\mathrm{R}}$ and $/_{\mathrm{L}}$. Thus, many recent formulations of CCG, such as [69] and [7], also make use of *modalized* (or "typed") slashes: each slash is equipped with not only a direction but also one of a finite set of *modalities*; rules can specify the particular slashes to which they apply. These modalized slashes are quite useful in describing languages with free or partially free word order (see [7]). In the reverse direction, there have been some attempts ([1], [65]) to do away with direction and have just one slash, with the CCG rules bearing the burden of specifying the order in which constituents combine. We will not have the space here to provide a detailed exposition of these theories, or to take up the question of whether having more slashes genuinely increases the expressive power of the system or merely provides a notational convenience. However, we will find that we can generalize our formal apparatus to one that allows arbitrary slashes without added complexity, and so we will take the liberty of doing so without explicitly giving further motivation.

We have now finished our whirlwind tour of the ideas of categorial grammar, giving the basic motivating ideas for the theory, numerous examples for particular rules, and a few of the possible extensions. Now it is time at last to give a formal framework for CCGs; this framework that can handle various types of rules, as well as modalized or non-directional slashes. The grammar system we will define here is a synthesis of ideas from [43, ch. 9] and [73]. It is also somewhat like the type-logical system of [45] in that it provides a general framework in which CCG rules can be defined. We will call our grammars *generalized combinatory categorial grammars* (GCCGs); the structures given here are much more general than any extant linguistic theory actually requires.

Let O be a finite set of symbols, which will be used to represent binary operators. For any finite set C (whose elements are called *primitive categories*), we let the *(O-)category space over C* be the set V obtained by taking the closure of C under the operations in O without any relations, just as in the case of classical CGs. (Thus, we can again formally construct V by setting $V_0 = C$, $V_{i+1} = V_i \cup (V_i \times O \times V_i)$, and $V = \cup_{i=0}^{\infty} V_i$.) For example, in the definition of classical CGs given above, O is just $\{/_{\mathrm{L}}, /_{\mathrm{R}}\}$. We will usually find it convenient to use $|$ as a variable ranging over the set O, and write $X|Y$ rather than $(X, |, Y)$.

If V_1 and V_2 are category spaces over primitive sets C_1 and C_2, then we have the natural notion of a *homomorphism* $\phi : V_1 \to V_2$, a map such that $\phi(X|Y) = \phi(X)|\phi(Y)$ for all $X, Y \in V_1, | \in O$. It is clear that a homomorphism is uniquely determined by its values on elements of C_1.

Let Ω be an arbitrary set disjoint from C, whose elements we call *variables*. (In general, we will be somewhat sloppy and may use the same letters to denote members of C and Ω; the meaning should be clear from context.) Let V be the O-category space over C, and let V' be the O-category space over $C \cup \Omega$; elements of V' are thus categories composed of primitive categories (in C), variables, or both. A *rule schema* is any statement of the form

$$A \to \omega, \qquad A \in V', \quad \omega \in (V')^{+}. \tag{28}$$

For example, if the symbols X, Y are in Ω and $/_{\mathrm{R}} \in O$, then $X \to X/_{\mathrm{R}}Y\ Y$ is a rule schema. If K is an element of C, then $X \to X/_{\mathrm{R}}\mathrm{K}\ Y$ is also a rule schema.

If R is a rule schema,

$$A \to A_1\ A_2\ \cdots\ A_n,$$

then a *ground instance* of R is any expression of the form

$$X \to X_1\ X_2\ \cdots\ X_n$$

such that there exists a homomorphism $\phi : V' \to V$ satisfying $\phi(Q) = Q$ for all $Q \in C$, and $\phi(A) = X$, $\phi(A_i) = X_i$ for each i. In other words, a ground instance of R is obtained by taking the schema R and replacing each variable with some category in V, where different occurrences of the same variable must be replaced by the same category. Notice that we are henceforth making a firm distinction between rules and schemata: a schema can involve variables; a rule is a ground instance of a schema and uses only categories over C.

Finally, suppose that we have some fixed O, C, Ω, and some set Φ of rule schemata (we refer to the set Φ as a *rule system*). Let P be the set of all ground instances of the rule schemata in Φ. Then we can treat the elements of P as productions over the infinite set of nonterminals V. If f is an assignment function mapping the terminals (elements of Σ) to finite subsets of V, and $S \in C$ is a distinguished element, then the sextuple $(O, C, S, \Omega, \Phi, f)$ is a *generalized combinatory categorial grammar* over the terminal alphabet Σ. The language generated by this grammar is simply the set $\{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$, using as rules all the elements of P together with the rules

$$X \to a \qquad (a \in \Sigma, \quad X \in f(a)).$$

By setting $O = \{/_{\mathrm{L}}, /_{\mathrm{R}}\}$, $\Omega = \{X, Y\}$, and $\Phi = \{X \to X/_{\mathrm{R}}Y\ Y, X \to Y\ X/_{\mathrm{L}}Y\}$, we recover the original definition of classical categorial grammars. In general, we can think of the set Φ (together with the sets O and Ω that are needed to define it) as specifying a theory of universal grammar, such as classical categorial grammar. An individual language's grammar within such a theory is then determined by a triple (C, S, f).

4 Generative capacity of categorial grammars

Now that we have defined the grammar frameworks that interest us, it is time to start obtaining some results. We should begin with a few clarifying comments on our motivating goals: we would like to obtain results about GCCG systems that are as general as possible. Although much work to date has naturally focused on the particular combinatory rules that have actually been proposed for natural language, we would like to begin to consider imaginable rule systems rather than merely plausible ones. The goal is not only the mathematical idea of greater generality but also more powerful explanation in the linguistic sphere (remember the discussion of evolutionary constraints in Section 1): if we can show that some classes of rules generate extremely restricted classes of languages while others generate excessively complicated languages, we will be in a better position to try to understand *why* human language is governed by some rules and not others, just as how Gold's Theorem (see Section 5) is often pointed to as a demonstration of the logical necessity of Universal Grammar. Conversely, if we find that many different rule systems all have equal generative power, we are allowed to be somewhat confused. Of course, a general classification and understanding of possible CCG rule systems (let alone of possible linguistic theories) is still a long way off, but we can at least adopt this mindset in driving our presentation.

We might point out that studying generative power is not the only possible way, within the CCG framework, to approach the question of why language behaves according to one set of rules and not another. Steedman ([67]) gives an intriguing argument, using ideas from combinatory logic ([24]), as to why natural language should prefer to use a certain basic repertoire of semantic operations, and insofar as categorial grammar supposes a tight fit between semantics and syntax, the range of syntactic operations is accordingly constrained. However, because his argument is not totally conclusive and still leaves the choice of rules somewhat underdetermined, it is worth investigating the question from other perspectives, so we will.

Another caveat is that we are still restricting our attention to weak generative capacity — looking only at languages as sets of strings, for the reasons outlined at the end of Section 2. There has, however, been some work on the strong generative capacity of various versions of categorial grammar; see [13], [72].

Now then, what does the (weak) generative power of CCG systems look like? Because CG rule schemata code for context-free rules, it intuitively seems that one should often expect them to generate context-free languages. If there are only finitely many different ground instances of rule schemata that ever actually occur in any derivations of the grammar G, then this will certainly be the case, since one can treat these ground instances as productions of a context-free grammar. But if G makes use of infinitely many different ground instances, then we may not get a context-free language.

Classical CGs, with just the rules (8), fall into the former category, and we thus have the following basic result:

Theorem 4 *[9] A language L (over a fixed Σ) can be generated by a classical categorial grammar if and only if L is a context-free language and $\epsilon \notin L$.*

(For now we will give an overview of the results; the proofs will follow later.)

The proof of the forward direction of the preceding theorem will entail showing that only finitely many ground instances of rule schemata are ever needed. One might guess that this idea could be generalized to other types of GCCG rule systems. We give one generalization which, to our knowledge, has not been noticed, but which there is no reason not to prove since it will turn out to be a straightforward extension of the proof of Theorem 4.

We first need to define the relevant class of rule schemata. Let C be a space of primitive categories, and let V be an O-category space over C. If $X, Y \in V$, we say that X is a *subtype* of Y if there is a sequence of categories $X = X_0, X_1, \ldots, X_n = Y$ so that, for each i, $X_{i+1} = X_i|Z$ or $Z|X_i$ for some $| \in O$ and some $Z \in V$. We say n is the *depth* of X as a subtype of Y. Thus, for a linguistic example, in $(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP}$, the subtypes of depth 1 are $\mathrm{S}/_{\mathrm{L}}\mathrm{NP}$ and NP; also, S and NP are subtypes of depth 2.

Suppose we have a set of variables Ω. Then a rule schema $A \to A_1 \cdots A_n$ will be called *depth-preserving* if the following holds: every variable that occurs as a subtype of A with some depth d must also occur as a subtype of some A_i with depth $\geq d$. The relevant result is then as follows:

Theorem 5 *If $G = (O, C, S, \Omega, \Phi, f)$ is a GCCG over the alphabet Σ, such that Φ is finite and every rule in Φ is depth-preserving, then the language generated by the grammar is context-free.*

The abstraction may be distracting, but a few examples should illustrate the cut of the result's jib. We will assume that O is the usual set $\{/_{\mathrm{L}}, /_{\mathrm{R}}\}$. The functional application rules (8) are both depth-preserving (X has depth 0 on the left side, and depth 1 on the right). The simple composition rules (10) are also depth-preserving. Hence, any GCCG using these four rule schemata (or any proper subset of them) generates a context-free language. Rule schemata such as

$$X/_{\mathrm{L}}X \to Y/_{\mathrm{R}}(Z/_{\mathrm{R}}X)\ Z/_{\mathrm{L}}Y$$

$$X/_{\mathrm{L}}\mathrm{I} \to Y/_{\mathrm{L}}X\ X\ \mathrm{J}/_{\mathrm{L}}X$$

(where $\mathrm{I}, \mathrm{J} \in V$ are specific categories) are also depth-preserving, so including these schemata also keeps us within context-free expressive power.

The crossed composition schemata (21) are also depth-preserving. In a sense, this contradicts the intuitive statement made in [69] that crossed composition is responsible for giving CCG greater than context-free expressive power. In fact, among the standard CCG rule schemata described in the preceding section, one needs type-raising or generalized (multi-layer) composition to generate non-context-free languages. (This observation is also made in [77, ch. 4].) For example, the composition rule

$$(X/_{\mathrm{R}}Z_1)/_{\mathrm{R}}Z_2 \to X/_{\mathrm{R}}Y\ (Y/_{\mathrm{R}}Z_1)/_{\mathrm{R}}Z_2$$

is not depth-preserving, since X has depth 2 on the left side but only depth 1 on the right side.

Of course, it is really an oversimplification to credit a single schema with generating non-context-free languages, since the language generated by a grammar depends on all of the rules used. Indeed, adding more rules can even lower the expressive power of the system (as measured by the Chomsky hierarchy). For example, if we start with some GCCG system and then add the rule schema $X \to Y\ Z$, where X, Y, Z are distinct variables, then any string of more than one terminal is now in the language, so we can only get regular languages.

The natural question now is whether we really do get non-context-free languages with suitable CCG rule schemata, or whether the added power is illusory. Here is an example of such a non-context-free language, inspired by the Dutch example (25). Suppose our set Φ of rule schemata consists of the functional application schemata (8), together with one schema from (23), namely

$$(X/_{\mathrm{L}}Z_1)/_{\mathrm{R}}Z_2 \to X/_{\mathrm{R}}Y\ (Y/_{\mathrm{L}}Z_1)/_{\mathrm{R}}Z_2.$$

Define a grammar with terminals $\{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}\}$, primitive categories $\{Q, R, S\}$, by setting

$$f(\mathrm{a}) = \{Q\} \quad f(\mathrm{b}) = \{R\} \quad f(\mathrm{c}) = \{(S/_{\mathrm{L}}Q)/_{\mathrm{R}}S\} \quad f(\mathrm{d}) = \{(S/_{\mathrm{L}}R)/_{\mathrm{R}}S\} \quad f(\mathrm{e}) = \{S\}$$

It can be shown that the language L generated by this grammar is not context-free. For a quick sketch of the proof, intersect L with the regular language $\{\mathrm{a}, \mathrm{b}\}^*\{\mathrm{c}, \mathrm{d}\}^*\mathrm{e}$, and then apply the homomorphism given by

$$\mathrm{a} \mapsto \mathrm{a} \quad \mathrm{b} \mapsto \mathrm{b} \quad \mathrm{c} \mapsto \mathrm{a} \quad \mathrm{d} \mapsto \mathrm{b} \quad \mathrm{e} \mapsto \epsilon$$

It it not too hard to check that the resulting language is the copy language $\{ww\}$ over $\{\mathrm{a}, \mathrm{b}\}$; since this language is not context-free, neither was the original L.

So if CCG is not restricted to context-free languages, what is the next reasonable class of languages that it can generate? Vijay-Shanker and Weir, in [73], define a type of CCG system that generates precisely the linear indexed languages. Their version of CCG does not quite fit into our framework, and so we will give an approximation of their result before discussing the differences.

Vijay-Shanker and Weir's result, paraphrased in terms of our concepts, is as follows.

Theorem 6 *Let $O = \{/_{\mathrm{L}}, /_{\mathrm{R}}\}$, and let $(O, C, S, \Omega, \Phi, f)$ be a GCCG. If Φ consists of finitely many schemata of the forms*

$$(\cdots(X|_1Z_1)\cdots|_nZ_n) \to (X/_{\mathrm{R}}Y)\ (\cdots(Y|_1Z_1)\cdots|_nZ_n) \tag{29}$$

$$(\cdots(X|_1Z_1)\cdots|_nZ_n) \to (\cdots(Y|_1Z_1)\cdots|_nZ_n)\ (X/_{\mathrm{L}}Y), \tag{30}$$

where X is a variable, each of Y, Z_i is either a distinct variable or a category over C, and each $|_i \in O$, then the language generated by the GCCG is a linear indexed language.

Thus, any CCG using a finite number of composition rules (including function application) will generate a linear indexed language. We will actually prove a slight generalization of the preceding result, but to state it, we will need a little more terminology. Let V be the O-category space over a set of primitive categories C. We define the *target* function $\tau : V \to C$, consistent with the definitions in the last section, recursively by $\tau(X) = X$ for $X \in C$, and $\tau(X|Y) = \tau(X)$. We also define the *arguments* of an element $X \in V$ to be the set given recursively as follows: X has no arguments if $X \in C$; and the arguments of $X|Y$ consist of Y and all the arguments of X. We now define a rule schema $A \to A_1 \cdots A_n$ to be *argument-depth-preserving* if the following three conditions hold:

- the target of A (viewed as a category over $C \cup \Omega$) is a variable and is also the target of exactly one A_i, and it does not appear anywhere else as a subtype of A or any A_j;
- every variable that is the target of some A_i and is *not* the target of A must also be a subtype of an argument of some A_j;
- any variable occuring as a subtype of an argument of A, with some depth d, is also a subtype of an argument of some A_i with depth $\geq d$.

Thus, for example, all the composition rules are argument-depth-preserving (with all the relevant depths equal to 0), even though they are not necessarily depth-preserving. A rule schema such as $X/_{\mathrm{R}}Y \rightarrow Y/_{\mathrm{L}}X$ is depth-preserving, but not argument-depth-preserving.

Theorem 7 *If $(O, C, S, \Omega, \Phi, f)$ is a GCCG, and Φ consists of finitely many argument-depth-preserving rules, then the resulting language is a linear indexed language.*

Our proof is a straightforward generalization of that given in [73] and [74] (and also outlined in [68]).

As mentioned earlier, there are a couple of differences between the CCG systems used by Vijay-Shanker and Weir and those we have defined. One is that they allow rule schemata in which a variable can be restricted to have a specific target. Although we have not included this extra notational complexity, one can check that the proof we give below still works in this case.

A more important difference is that Vijay-Shanker and Weir allow the function f to assign categories to the empty string ϵ, in addition to terminal symbols. It is also straightforward to adjust our proof to handle this. However, under this definition of CCGs, they show not only that every language generated by a CCG (with composition rule schemata) is generated by an LIG but also that the converse is true. (The construction is too lengthy to warrant including here; see [73] for details.)

Unfortunately, the proof given by Vijay-Shanker and Weir that every linear indexed language is generated by a CCG depends on assigning a large number of categories to ϵ, so it does not go through if we prohibit ϵ from receiving categories. As noted by Weir and Joshi in commenting on these results ([75]), linguistic applications of CCG do not assign categories to ϵ. It seems rather undesirable to introduce this ad-hoc allowance, which both complicates the formalism and renders categorial grammar less appealing as a linguistic theory. This naturally raises the question of whether excluding ϵ gives a proper subset of the linear indexed languages (aside from the obvious property that we cannot generate any L with $\epsilon \in L$), which appears to be an open problem. This prospect may or may not raise questions for the viability of this form of CCG, but at any rate, it is time to move on.

If we allow our GCCG rules to be completely arbitrary, we are not even restricted to the linear indexed languages. For one (rather artificial) example, consider the GCCG with $O = \{|\}$, $C = \{Q, R, S\}$, $\Omega = \{X, Y\}$, and the rules

$$S \rightarrow X|S\ X|S\ X|S \qquad (X|Y)|S \rightarrow X|S\ Y|S.$$

If our terminal symbols are a, b, assigned to categories $Q|S, R|S$ respectively, then we have (for example) $(Q|R|Q|Q|R)|S \overset{*}{\Rightarrow}$ abaab, under any parenthesization of the Q's and R's. In general, if X is a category not having S as a subtype, then $X|S \overset{*}{\Rightarrow} w$ for a unique string w, "encoded" by the order of the Q's and R's in S. It follows that our grammar generates precisely the double-copy language $\{www\}$.

In fact, it is an exercise to construct a GCCG with finitely many rules that mimics any given unrestricted rewrite system, even with only one operation in O; this implies that GCCGs can generate any recursively enumerable language, which is far too broad. However, recall that we do not want to think of GCCG itself as a model of universal grammar; rather, any particular set Φ of rule schemata gives us a model of universal grammar, which in turn determines a class of languages.

As we have just seen, it is easy to concoct ad-hoc rule systems to give us arbitrary (recursively enumerable) language, so in order to obtain results we can care about, we should restrict our attention to rules that are plausibly linguistically relevant. We can give examples of some rules actually proposed in the literature that still have the effect of allowing non-linear-indexed languages. For one, note that the statement of Theorem 6 allows only finitely many composition rules; a natural extension of the CCGs we have defined would be to allow GCCGs whose rule schemata consist of infinitely many composition rules. As pointed out in [75], however, this allows us to generate non-linear-indexed languages. The example given there (also in [74]) is essentially as follows: consider a GCCG over $O = \{/_{\mathrm{L}}, /_{\mathrm{R}}\}$ with all possible mixed composition rules (i.e. those of the forms (29), (30)). Then we define f as follows:

$$f(\mathrm{a}_1) = \{A_1\} \qquad f(\mathrm{a}_2) = \{A_2\} \qquad f(\mathrm{b}_1) = \{B_1\} \qquad f(\mathrm{b}_2) = \{B_2\}$$

$$f(\mathrm{c}_1) = \{(((S/_{\mathrm{L}}A_1)/_{\mathrm{R}}D_1)/_{\mathrm{R}}S)/_{\mathrm{L}}B_1\} \qquad f(\mathrm{c}_2) = \{(((S/_{\mathrm{L}}A_2)/_{\mathrm{R}}D_2)/_{\mathrm{R}}S)/_{\mathrm{L}}B_2\}$$

$$f(\mathrm{d}_1) = \{D_1\} \qquad f(\mathrm{d}_2) = \{D_2\} \qquad f(\mathrm{e}) = \{S\}$$

The language L generated by this grammar is rather complicated, but one can show that its intersection with the regular language

$$\{\mathrm{a}_1\}^*\{\mathrm{a}_2\}^*\{\mathrm{b}_1\}^*\{\mathrm{c}_1\}^*\{\mathrm{b}_2\}^*\{\mathrm{c}_2\}^*\mathrm{e}\{\mathrm{d}_2\}^*\{\mathrm{d}_1\}^*$$

consists of all strings of the form

$$\mathrm{a}_1^{n_1}\mathrm{a}_2^{n_2}\mathrm{b}_1^{n_1}\mathrm{c}_1^{n_1}\mathrm{b}_2^{n_2}\mathrm{c}_2^{n_2}\mathrm{e}\mathrm{d}_2^{n_2}\mathrm{d}_1^{n_1}$$

for $n_1, n_2 \geq 0$. This set can be shown not to be a linear indexed language, by the pumping lemma for LIL's ([71], [48]). Since the class of LIL's is closed under regular intersection (it is an AFL, [76]), then the original language L is not an LIL.

If we have only finitely many schemata, documentation of non-linear-indexed languages in the literature is extremely sparse. Weir ([74], also [75]) claims that a conjunction rule can generate non-linear-indexed languages but gives no example. It also seems that type-lifting rules should be able to generate non-linear-indexed languages, but examples are not forthcoming. Here we will sketch an example for a somewhat lifting-like rule that comes from [35]. Let Φ consist of the functional application schemata and mixed composition (23) for $n \leq 2$, as well as the following schema:

$$(X/_{\mathrm{L}}Y)/_{\mathrm{R}}(X/_{\mathrm{L}}Z) \to Z/_{\mathrm{R}}Y$$

Now define f by

$$f(\mathrm{a}_1) = \{A_1\} \qquad f(\mathrm{a}_2) = \{A_2/_{\mathrm{R}}A_1\} \qquad f(\mathrm{a}_3) = \{A_3/_{\mathrm{R}}A_2\}$$

$$f(\mathrm{a}_4) = \{A_4/_{\mathrm{R}}A_3\} \qquad f(\mathrm{a}_5) = \{(S/_{\mathrm{L}}A_4)/_{\mathrm{R}}S\} \qquad f(\mathrm{s}) = \{S\}$$

When the language generated by this categorial grammar is intersected with the language $\{\mathrm{a}_1\}^*\{\mathrm{a}_2\}^*\{\mathrm{a}_3\}^*\{\mathrm{a}_4\}^*\{\mathrm{a}_5\}^*\mathrm{s}$, the resulting language is $\{\mathrm{a}_1^n\mathrm{a}_2^n\mathrm{a}_3^n\mathrm{a}_4^n\mathrm{a}_5^n\mathrm{s}\}$, which is not linear indexed.

These kinds of findings may pose a problem for constraining the expressive power of CCG, although it still remains to be determined whether or not type-lifting can really take us out of the linear-indexed class. Meanwhile, one way that has been proposed to bridge the gap is lexicalization — constraining each rule to a finite number of possible applications, which would then be captured by the assignment function f. For example, in [68], Steedman hints at lexicalizing type-lifting, so that instead of a lifting schema, some collection of NP's (say) are simply assigned the type $\mathrm{S}/_{\mathrm{R}}(\mathrm{S}/_{\mathrm{L}}\mathrm{NP})$. However, this approach poses further problems for linguistic description, since we would like to be able to type-lift any NP in order to let it participate in right-node-raising constructions (18).

One upshot of all this is that the book is far from closed on the formal expressive power of categorial grammars. Even if we restrict our attention to versions of the formalism that have been reasonably well fleshed-out as potential descriptions of natural language, interesting and relevant questions about expressive power are still open. Another observation to draw away is that, as Theorems 5 and 7 suggest, there are many possible sets of rules that give comparable degrees of expressive power. This mitigates against our preferring one set of rules rather than others on purely formal grounds; on the flip side, it also suggests that the tendency for several seemingly very different formalisms to give the same class of languages, such as the linear-indexed and related systems studied by Vijay-Shanker and Weir, should perhaps be taken with a grain of salt.

To summarize our results so far: Theorem 5 shows that (finitely many) depth-preserving rule schemata, such as those of classical CG, can only give context-free languages. Theorem 7 shows that argument-depth-preserving rule schemata, such as generalized mixed composition, can only give linear indexed languages. We also have seen reasons to believe that other CCG rules may take us out of the linear indexed class, and certainly GCCGs taken as a whole can wildly overgenerate.

Now that we have given an overview of some of the results about the generative power of categorial grammars, let us go ahead and prove them. The proofs are presented here, rather than left entirely for the appendix, in order to give an indication of the kinds of methods that are used in this area. The general technique is straightforward: to show that a class of rules generates (say) only linear indexed languages, one simply performs a reduction, transforming any given categorial grammar into a linear indexed grammar that gives the same language. The rub is that sometimes the desired reduction is possible and sometimes not.

We begin with Theorems 4 and 5. As indicated, the reduction of depth-preserving CCGs to context-free grammars is fairly straightforward: one only has to show that the derivations of G altogether use only finitely many different ground instances of the rule schemata.

Proof of Theorem 4: First suppose that L can be generated by a classical categorial grammar (C, S, f). It is clear from the definitions that $\epsilon \notin L$. To show that L is a context-free language, it suffices to show that there are only finitely many categories appearing anywhere in any S-parse tree of the grammar. Then it will follow that only finitely many ground instances of the application rules are used in any such tree, which means that we can form a context-free grammar for L using these ground instances, together with the rules $X \to a$ ($X \in f(a)$), as productions.

Let K be the set of all subtypes of any category assigned to an element of Σ by f. Then we claim that, in any S-parse tree, all the categories occurring belong to K. To see this, we work by induction up the tree. It is clear that each category immediately dominating a terminal a belongs to K (since this category is in $f(a)$). For the induction step, we need to show that with each application of a rule $X \to X/_{\mathrm{R}}Y\ Y$, the known $X/_{\mathrm{R}}Y \in K$, $Y \in K$ (given by the induction hypothesis) implies $X \in K$. But this is certainly true, since X is a subtype of $X/_{\mathrm{R}}Y$, and subtypes of elements of K are in K. Similarly for the rule $X \to Y\ X/_{\mathrm{L}}Y$, and so the induction step holds. Since K is finite, this completes the proof that L is context-free.

The proof of the converse — that every ϵ-free context-free language is given by a classical CG — is rather lengthy, and is not really essential to our immediate concerns (since what we have already shown effectively tells us that classical CGs are not an adequate model for natural language). We will only state it as a separate lemma here, but we include the proof in the appendix, since the result will be referenced in subsequent sections.

Lemma 8 *Any context-free language not containing ϵ can be generated by a classical categorial grammar.*

This granted, we are done. □

Proof of Theorem 5: As in the proof of Theorem 4, it suffices to show that there are only finitely many categories that can occur in any S-parse tree of our grammar G (and it then immediately also follows that only finitely many ground instances of the rules of Φ can be used). So let d be the maximum depth to which any variable appears in the left side of any rule schema in Φ. Let K be the set of all subtypes of any type assigned to a terminal symbol by f, together with any types over C that occur anywhere in a rule of Φ; and let K' be the set of all types with the property that every subtype of depth at least d is in K. Notice that K' is finite. Indeed, it is equal to the set K_d obtained in the course of the usual category space construction ($K_0 = K, K_{i+1} = K_i \cup (K_i \times O \times K_i)$).

Now we just need to prove that, in every S-parse tree under G, every category that appears is in K'. This is done just as in the previous proof. We show by induction, working up the tree, that every nonterminal used is in K'; this is immediate for the nodes immediately above terminals, since the categories appearing there are in $K \subseteq K'$. For the induction step, we need to show that whenever we have a ground instance $X \to X_1 \cdots X_n$ of a rule schema $A \to A_1 \cdots A_n$, if $X_1, \ldots, X_n$ are in K' then so is X. Well, consider any subtype Y of X of depth at least d. Either it arises from a non-variable in A (i.e. a subtype of A that is a category over C), in which case it is already in K; or it arises from the instantiation of a variable B in A. In the latter case, the variable B is a subtype of A of some depth d_B; it is replaced in our ground instance by some type Z; and Y is a subtype of Z, of some depth d_Y. The depth of Y as a subtype of X is $d_B + d_Y \geq d$. By the assumption that our rules are depth-preserving, B must also occur in some A_i to depth at least d_B, so Z is a subtype of X_i of depth at least d_B and hence Y is a subtype of X_i of depth $\geq d_B + d_Y \geq d$. Therefore $Y \in K$. This shows that $X \in K'$, which is what we need, and the proof is done. □

The proof of Theorem 7 is next. (Theorem 6 is just a special case.) This uses analogous ideas, only one has to be slightly cleverer in order to transform argument-depth-preserving rule schemata into LIG productions.

Proof of Theorem 7: The first step imitates the proof of Theorem 5. Let K be the set of all subtypes of categories assigned to terminals by f, together with all categories over C appearing as subtypes in any rule schema in Φ. Let d be the maximum depth to which any variable appears on the left side of a rule schema. Let K' be the set of all categories such that every subtype of depth at least d is in K. Then K' is a finite set. We claim that, in every S-parse tree from our grammar, every category that appears has all of its arguments in K'. The proof is by induction up the tree, exactly as in the proof of Theorem 5; the third condition in the definition of argument-depth-preserving ensures that the induction step succeeds.

Now, define a *semi-ground instance* of an argument-depth-preserving rule schema $A \rightarrow A_1 \cdots A_n$ to be a schema $B \rightarrow B_1 \cdots B_n$, where B, B_i are categories over $C \cup \Omega$, with the following property: there exists a homomorphism ϕ from the category space over $C \cup \Omega$ to itself, such that $\phi(A) = B$ and $\phi(A_i) = B_i$; moreover $\phi(Q) = Q$ for $Q \in C$, and ϕ maps every variable to a category over C, *except* that $\phi(\tau(A))$ is still a variable. Thus, a semi-ground instance is almost the same as a ground instance, except that the target of A remains a variable rather than being mapped to a category over C.

We claim that we can replace each rule schema R of Φ by finitely many semi-ground instances of R, without changing the language generated by the grammar — that is, only finitely many different semi-ground instances of schemata are actually used in any derivation. This is not hard to see: in any application of a rule schema $A \rightarrow A_1 \cdots A_n$, all the argument categories appearing on either side of the rule must be from the finite set K' (by the result from two paragraphs ago), and all the target categories except for $\tau(A)$ must also be instantiated as elements of K' since they also appear in argument categories; thus, every variable in the rule other than $\tau(A)$ can take on only finitely many values. Thus, we may now assume that every rule schema R has exactly one variable, and that moreover that variable appears as the target of the left side of R, the target of one term on the right side, and nowhere else.

Now we are ready to construct our linear indexed grammar. Let the set of nonterminals be C, and let the set of stack symbols I be $O \cup K'$. Write $W = C \times I^*$ as usual, and let V' be the set of categories over C whose arguments all belong to K'. Define the "encoding" function $e : V' \rightarrow W$ as follows: any category in V' can be uniquely written $(\cdots((X|_1Y_1)|_2Y_2)\cdots|_nY_n)$ for $X \in C, |_i \in O$, and $Y_i \in K'$; then

$$e(\cdots((X|_1Y_1)|_2Y_2), \ldots |_nY_n) = X[|_1Y_1 \cdots |_nY_n].$$

We have the following LIG rules:

- if $A \rightarrow A_1 \cdots A_n$ is a rule schema in Φ, where $\tau(A) = \tau(A_i)$ is the variable X, then write $A = (\cdots(X|_1Y_1)\cdots|_nY_n), A_i = (\cdots(X|'_1Y'_1)\cdots|'_nY'_n)$, and for each category $Q \in C$ we then use the LIG rule

$$Q[\circ\circ|_1Y_1 \cdots |_mY_m] \;\rightarrow\; e(A_1)\cdots e(A_{i-1})\, Q[\circ\circ|'_1Y'_1 \cdots |'_mY'_m]\, e(A_{i+1})\cdots e(A_n); \qquad (31)$$

- if $X \in f(a)$, then we also have the rule $e(X) \to a$.

It is now immediate that $X_1 \cdots X_n \Rightarrow Y_1 \cdots Y_m$ in the GCCG (where the X_i, Y_j are categories in V') if and only if $e(X_1) \cdots e(X_n) \Rightarrow e(Y_1) \cdots e(Y_m)$ in the linear indexed grammar we have constructed. Applying this to each step of a derivation gives us, for any terminal string w, that $S \overset{*}{\Rightarrow} w$ in the GCCG if and only if $S[\,] \overset{*}{\Rightarrow} w$ in the LIG, so we are done. □

Before departing from this brief glance at classes of CCG rule schemata, we should mention two topics that are worth consideration in the course of further investigation. One is whether the size of the operation set O actually makes any difference in the expressive power of a CCG system. Of course this is not an immediately well-defined question, because we do not know how to talk about rule systems without fixing O. But all the classes of rule systems we have considered here seem to give expressivity results that are independent of the number of operations. Thus, Theorem 5 tells us that no matter how many slashes we have, depth-preserving rules give us only ϵ-free, context-free languages, and the proof of Lemma 8 given in the appendix shows that any such language can be achieved by using only one slash (say, $/_{\mathrm{L}}$). In discussing modalized slashes in [7], Baldridge shows that they do not actually add anything to the expressive power of his CCG formalism, although we cannot use his proof here because it requires a framework with target-restricted rules. As mentioned above, if we consider GCCGs in full generality (with Φ finite), we get the class of all recursively enumerable languages, and restricting O to be a singleton again does not change this. It would be interesting to generalize results of these sorts.

Next, we mention one criterion for CCG rules that intuitively seems relevant and that has been missing thus far. Many of the rule schemata we have proposed are quite ridiculous. For example, a rule like $X/_{\mathrm{R}}Y \to Y/_{\mathrm{L}}X$ seems unnatural; an expression that forms an X if a Y is added to it should not also form a Y if an X is added to it. What we, as decent human beings, might expect from a combinatory rule schema is that the elements on the "numerator" of one side of the rule should also be on the numerator of the other side, and similarly for the denominators. We can make this algebraically precise. If C is a set of primitive categories and V the category space over it, let U be the free abelian group generated by the elements of C, written multiplicatively; so elements of U are all of the form $X_1^{e_1} X_2^{e_2} \cdots X_n^{e_n}$, where the X_i are in C and e_i are arbitrary integers, and multiplication of two elements is given by addition of corresponding exponents. Now define a map $\mu : V \to U$ by $\mu(X) = X$ for $X \in C$, and $\mu(X|Y) = \mu(X)\mu(Y)^{-1}$.

Now, we say that a GCCG rule schema $A \to A_1 \cdots A_n$, with A, A_i categories over $C \cup \Omega$, is *balanced* if $\mu(A) = \mu(A_1) \cdots \mu(A_n)$ in U. Then the function application, composition, and type-lifting rules are all balanced, but many others that we can imagine are not, such as $X/_{\mathrm{R}}Y \to Y/_{\mathrm{L}}X$. The property of being balanced seems to be a kind of baseline for reasonableness of a hypothesized GCCG rule. We leave as an open direction of study the question of whether using only a finite set of balanced rules still allows us to generate all recursively enumerable languages, or whether it gives us at least some restrictions (say, the constant growth property). It is worth pointing out, though, that unbalanced rules have been proposed, such as the substitution rule of [66].

After having looked at the generative power of some CCG systems, it seems appropriate to talk at least briefly about the analogous results for other linguistic theories, to see what the landscape is like. Unfortunately, most theories suffer from the unsurprising problem that the richer the description given, the more complicated the theory is to write down formally, and therefore the harder it is to obtain any results about. Transformational theories are so widespread and diverse (and, quite often, not fully fleshed out) that it is difficult to make any general statements about them. However, we should at least mention one seminal bit of work in this area, by Peters and Ritchie ([51], [50]). They give a general definition for transformational grammars, in which a context-free "base" grammar is used to generate parse trees; these trees are then operated on by deletion, substitution, and adjunction transformations whose applicability is determined by structural conditions on the tree. Peters and Ritchie show that every recursively enumerable language is generated by some such transformational grammar.

Of course, this is not at all a conclusive dismissal of transformational grammars as overgenerating, because any particular transformational linguistic theory is bound to be much more specific than the framework Peters and Ritchie give; their work serves as more of a cautionary note. On a more optimistic note for the expressivity of transformational grammars, Michaelis ([44]) has shown that a certain formalism for minimalist grammars, introduced by Stabler in [63], generates only languages satisfying Joshi's mild context-sensitivity requirements.

5 Learnability theory

The main motivating concept of learnability theory, as applied to natural language, is as follows: the class $\mathcal{L}^*$ of possible human languages must be small enough that some one "learner" is capable of learning any language. Indeed, any normal human child has the ability to learn any language. Once we have a suitable formalization of the notion of "learning," this idea can be used to evaluate theories of language; a theory that predicts an unlearnable class $\mathcal{L}$ is insufficiently restrictive. The theory of language learnability is logically almost entirely separate from formal language theory, although, in practice, ideas from the latter area are often used to delimit possible classes of languages to test for learnability.

The model of learning that we will use is among the simplest available; it comes from Gold's foundational article [31] and is formally referred to as *identification in the limit from positive data*. Our definitions come roughly from [43], but we will diverge in one important respect from much of the literature on the subject in that our model studies abstract learning functions rather than (computable) algorithms. This is done for formal simplicity, but it will not really make much difference, because in each case where we have a learnable class, we will give a computable learner.

As usual, consider a fixed alphabet Σ. For any nonempty language L, a *text* of L is an infinite sequence $w_1, w_2, \ldots$ of strings of L such that every element of L appears at least once. A *learner* for some class $\mathcal{L}$ of languages over Σ is a function $\Lambda : (\Sigma^*)^* \to \mathcal{L}$, that is, from the set of finite sequences of strings to $\mathcal{L}$. If L is a language in $\mathcal{L}$, we say that Λ *learns* L if the following holds: for every text $w_1, w_2, \ldots$ of L, there is some N such that

$$\Lambda(w_1, w_2, \ldots, w_n) = L \qquad \text{for all } n > N. \tag{32}$$

The idea is that the human child (modeled by Λ) successively hears strings from the language; after hearing any number n of strings, it has a current guess $\Lambda(w_1, \ldots, w_n)$ as to what the language is, and in order to learn the language, the child must eventually guess the correct language and stay there.

We say that the learner Λ *learns the class* $\mathcal{L}$ if it learns every language in $\mathcal{L}$, and that $\mathcal{L}$ is *learnable* if there exists a learner that learns $\mathcal{L}$. (All of these definitions only make sense for nonempty languages, so from now on it will be assumed that the empty language is excluded from $\mathcal{L}$, although we may not state this technicality explicitly.)

A couple of very simple examples are in order. If $\mathcal{L}$ is the class of all finite languages, then let $\Lambda(w_1, \ldots, w_n) = \{w_i \mid 1 \leq i \leq n\}$: the learner simply guesses the language consisting of all strings it has heard so far. Then Λ learns every finite language L: by definition of a text, every string in L equals some w_i, so $\{w_i \mid 1 \leq i \leq n\}$ is all of L when n is large enough. Hence, the class of all finite languages is learnable. Of course, since this algorithm only guesses finite languages, it will not be able to learn any class containing an infinite language (such as any human language).

If $\mathcal{L}$ contains only finitely many languages, then it is learnable. Indeed, just define Λ by letting $\Lambda(w_1, \ldots, w_n)$ be a minimal element of the set

$$\{L \in \mathcal{L} \mid w_1, \ldots, w_n \in L\}$$

(ordered under inclusion $L \subseteq L'$). To see that this learns any $L \in \mathcal{L}$, observe that there are finitely many $L'_j \in \mathcal{L}$ with $L \not\subseteq L'_j$. For each such L'_j, there is some $w'_j \in L$ with $w'_j \notin L'_j$. For any given text, once n is sufficiently large, all w'_j have occurred in the text. So the set $\{L' \in \mathcal{L} \mid w_1, \ldots, w_n \in L'\}$ does not contain any L'_j, hence it consists entirely of languages L' with $L \subseteq L'$ — and at this point the learner guesses its unique minimal element L. This shows that $\mathcal{L}$ is learnable.

A natural question to ask is whether any of the levels of the Chomsky hierarchy is learnable, since these are the most natural classes of languages to which we have reference. As we shall see momentarily, none of these classes is learnable. However, before going further, it seems appropriate to discuss some features of our model of learning and place it in the context of other possible models, as well as motivating some properties of the model that might initially be mysterious.

There certainly are other models of language learnability, and Savitch ([60]) gives a helpful survey. One clearly rough feature of our formalization is the definition of a text. For example, we have assumed that the text contains only positive data (i.e. the information that certain strings are in the language, rather than that certain strings are not in the language). This is based on the empirical finding that human children are exposed to very little negative data (part of the well-known *logical problem of language acquisition*), and that they often fail to absorb negative data such as parents' corrections to their own ungrammatical utterances. (See [6], [21], [36].)

Aside from the empirical reasons, there are formal reasons to define texts as we do. In [31], Gold also considers learning from an "informant" — either by allowing the learner to periodically propose strings and find out whether or not they are in the language L, or by having the text contain every string of Σ^*, together with an indicator as to whether or not it is in L (these two models actually give equivalent definitions of learnability). It turns out that outrageously large classes of languages are learnable in this model, which makes it not particularly useful for our purposes. Also, it should be clear why we require the text to contain every string of L at least once. If we simply allow the text to be any sequence of strings of L, then we have no way of differentiating a text of L from a text of some $L' \subset L$, so any class that contains two such languages is unlearnable.

The other component of our model is the definition of learning. Notice that we require the learner to reach the correct language, but it does not have to "know" when its guess is correct — that is, it always holds out the prospect of changing its guess in the future. This seems reasonable, since actual humans do not pick a point at which to stop acquiring their native language.

Looser definitions of learning are possible. Feldman ([26]) provides a model in which the learner Λ learns L if, for every $w \in L$, all but finitely many of the learner's guesses are languages containing w; every language other than L is guessed only finitely many times; and the correct language L is guessed infinitely many times. Other models allow the learner to be successful if it has come reasonably close to the target language (under some metric), or assume a probability distribution on L from which successive strings of the text are drawn, and require the learner to determine the correct language by some particular time for "most" texts ([70]). Although some of these models suffer from the same formal deficiency as informant learning — implausibly large classes of languages become learnable — the point should be made that many models are possible.

As mentioned above, one often studies "effective learning," where the learning function has to be computable. In this framework, one assumes a fixed grammar system, which maps some set of grammars $\mathcal{G}$ into the class of languages $\mathcal{L}$. The learning function Λ is required to be computable, and it takes values in $\mathcal{G}$ rather than $\mathcal{L}$ (grammars are finite objects and so can be the output of a computation; languages cannot). Λ learns L if it fixates on any grammar that generates L.

In the effective learning context, one can further constrain learnability; for example, computational learning theory often requires the learning algorithm to operate in polynomial time ([70]). Although this sort of consideration is certainly relevant to making predictions about the size of the class $\mathcal{L}$, it is beyond the scope of our current concerns. Actually, it is possible to use ideas from complexity theory even without explicitly requiring the computability of the learning function. For example, [79] discusses a constraint in which the number of wrong guesses the learner makes is polynomially bounded in terms of the size of the grammar. We are restricting attention here to the framework of identification in the limit from positive data because of its relative formal tractability, but the same questions we will study can and should be studied for other frameworks.

Let us now return to the Gold model of learning from text. As mentioned before, none of the classes of the Chomsky hierarchy is learnable. In order to show this, we give the following useful characterization of learnability, from Angluin ([4]):

Lemma 9 *A countable class $\mathcal{L}$ of nonempty languages is learnable if and only if, for each $L \in \mathcal{L}$, there is a finite "telltale" subset $T \subseteq L$ such that L is minimal in $\{L' \in \mathcal{L} \mid T \subseteq L'\}$.*

The countability restriction is not much of an impediment, since any learnable class must be countable. Indeed, given the learner Λ, every language in the class must equal $\Lambda(w_1, \ldots, w_n)$ for some finite sequence $w_1, \ldots, w_n$, and there are only countably many such sequences.

The proof of Lemma 9 is fairly simple: If each L does have such a finite subset T, then we define a learner that basically chooses a minimal language consistent with all the data so far; for the converse, if some language L has no such subset, then for any putative learner Λ we can produce a text that leaves Λ repeatedly thinking it is learning some language smaller than L. Indeed, Lemma 9 is a gratifying result for basic intuitions about what learning should require. It essentially says that no class is learnable unless it is learnable in the most obvious way, namely by always guessing the smallest language compatible with the data so far.

Proof: Suppose Λ is a function that learns $\mathcal{L}$. Let $L \in \mathcal{L}$, and suppose for the sake of contradiction that, for every finite $T \subseteq L$, there exists another language of $\mathcal{L}$ containing T and properly contained in L. Construct a text $w_1, w_2, \dots$ for L as follows. First let $u_1, u_2, \dots$ be any text for L. Then:

1. Since Λ learns L, there exists some n such that $\Lambda(u_1, \dots, u_n) = L$. Let $w_i = u_i$ for $i = 1, \dots, n$.

2. Suppose that $w_1, \dots, w_n$ have been determined. By hypothesis, there exists some $L' \in \mathcal{L}$ properly contained in L and containing $w_1, \dots, w_n$. Let $v_1, v_2, \dots$ be a text for L' such that $v_1 = w_1, \dots, v_n = w_n$. Since Λ learns L', we have $\Lambda(v_1, \dots, v_m) = L'$ for some $m > n$. So put $w_i = v_i$ for $n < i \leq m$. These strings are all in L, since $L' \subset L$. Thus $\Lambda(w_1, \dots, w_m) = L'$.

3. Next, choose the smallest k such that u_k has not occurred among $w_1, \dots, w_n$ (if there is such a k), and let $w_{n+1} = u_k$.

4. Iterate steps 2 and 3 indefinitely.

We thus construct an infinite sequence of strings from L, and step 3 ensures that every u_k eventually occurs, so that we do have a text (i.e. every string of L appears). On the other hand, step 2 ensures that there are infinitely many initial segments $w_1, \dots, w_n$ such that $\Lambda(w_1, \dots, w_n)$ is a proper subset of L rather than L. Thus, Λ fails to learn L. This is our needed contradiction.

For the converse, suppose each $L \in \mathcal{L}$ contains some finite set T such that L is minimal among all languages of $\mathcal{L}$ containing T. We also need our countability condition; let $L_1, L_2, \dots$ be an enumeration of the languages in $\mathcal{L}$. We now define Λ as follows: If the set $\{L' \in \mathcal{L} \mid w_1, \dots, w_n \in L'\}$ has a minimal element (under inclusion), then $\Lambda(w_1, \dots, w_n)$ be such a minimal element L_i, chosen from among all minimal elements so that i is as small as possible. Otherwise $\Lambda(w_1, \dots, w_n)$ can be an arbitrary language of $\mathcal{L}$. We need to show that this Λ learns any $L_i \in \mathcal{L}$. Well, let T be the telltale subset associated with L_i, and let $w_1, w_2, \dots$ be any text for L_i. When n is large enough, say $n > N$, then $w_1, \dots, w_n$ contains all the elements of T. Any language $L' \in \mathcal{L}$ that contains $w_1, \dots, w_n$ certainly contains T, so the set of such languages has a minimal element (namely L_i). Therefore, for all sufficiently large n, we have that $\Lambda(w_1, \dots, w_n) = L_j$ for some $j \leq i$.

We will be done if we can show that, for each $j < i$, $\Lambda(w_1, \dots, w_n) \neq L_j$ when n is sufficiently large (because then the only possibility remaining for all large enough n is the desired $\Lambda(w_1, \dots, w_n) = L_i$). Well, if $\Lambda(w_1, \dots, w_n) \neq L_j$ for all $n > N$, then we are done.

Otherwise, choose $n > N$ with $\Lambda(w_1, \ldots, w_n) = L_j$. Since L_j is minimal among the languages of L containing $w_1, \ldots, w_n$, then $L_i \not\subseteq L_j$. So L_i contains a string not in L_j, which must be manifest as w_{n_j} for some integer n_j. Hence, whenever $n \geq n_j$, we have $\Lambda(w_1, \ldots, w_n) \neq L_j$. This completes the proof. □

Now we are in a position to prove the famous result known as Gold's Theorem.

Theorem 10 *[31] If $\mathcal{L}$ contains all finite languages and at least one infinite language, then $\mathcal{L}$ is not learnable.*

Proof: If $\mathcal{L}$ is uncountable, then as we have already seen it is unlearnable, so suppose it is countable. Let L be the infinite language. According to Lemma 9, there must be some finite subset T so that L is minimal among languages in $\mathcal{L}$ containing T. But T is itself a language in $\mathcal{L}$ containing T and properly contained in L — contradiction. □

Of course, there is nothing terribly special about classes satisfying the hypotheses of Theorem 10, and one can use the same method to prove that other classes are unlearnable. For example, the class of all L such that $\Sigma^* - L$ is finite is unlearnable.

Since (say) all finite languages are regular, Theorem 10 implies that the class of regular languages is unlearnable. Also, if any class $\mathcal{L}$ is learnable, then so is every subset of $\mathcal{L}$ (use the same learner), so we immediately have

Corollary 11 *None of the levels of the Chomsky hierarchy is learnable.*

□

This result has been taken by Gold, and others since, as a demonstration that none of these levels is an adequate choice of $\mathcal{L}$, so that the class of human languages must be described by some other sort of restrictions.

This result, announced in 1967, was at first taken to constitute a major setback for the relevance of learnability theory to natural languages, since classes of languages worthy of consideration other than those of the Chomsky hierarchy were less readily available. However, the area was revitalized by Angluin's 1980 discovery ([5]) of other nontrivial learnable classes of languages. Work in the late eighties by Buszkowski, such as [12] and [14], showed that adding simple restrictions to the categorial grammar formalism could produce learnable classes, leading to the results we present here. Our material on the learning of categorial grammars will be taken from Kanazawa's survey book [43], except where noted otherwise. We have significantly tightened some of the proofs, but the general structure preserves the spirit of Kanazawa's treatment. One might expect that we would use Lemma 9 to prove learnability, but in fact the existence of tell-tale subsets for classes of languages is not always an easy thing to prove, so we will instead go about our business by constructing learning algorithms.

In order to demonstrate how certain classes of categorial grammars can be learned (the main goal of the next section), we first need a slight detour into learning other kinds of objects. We have looked so far at studying languages of strings — subsets of Σ^* — but really, this has been quite irrelevant. The foregoing results, such as Lemma 9 and the learnability of finite classes, hold equally well when the objects being identified are subsets of some arbitrary countable set Δ, and thus $\mathcal{L}$ is a class of subsets of Δ. For example, we could speak of "learning" a set of integers, or learning the names of Uncle Ned's goldfish, rather than learning a language, with all the same formal implementation.

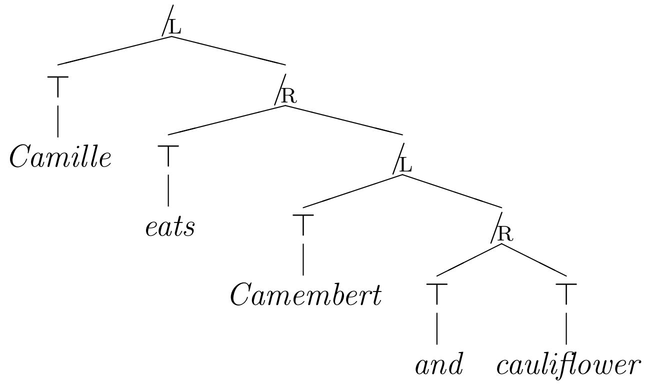
In particular, we can let Δ be the set of trees (suitably defined) generated by a grammar, rather than merely the set of strings generated by that grammar. Following the terminology used by Kanazawa, we then think of the corresponding learning problem as one of "learning from structures." Although we have heretofore avoided burdening our poor brains with the additional complexity of trees, we will see that, in the context of CCG systems, starting with learning from structures will help us get a handle on the more directly relevant problem of learnability of string languages.

We need to define the structures under consideration. Suppose we have a fixed GCCG $G = (O, C, S, \Omega, \Phi, f)$ over the alphabet Σ. We have the notion of an S-parse tree, as usual: a (labeled, ordered) tree, such that

- the leaves are labeled with terminals;
- the other nodes are labeled with categories over C;
- if a node has a label X and its daughters' labels, in order, are categories $X_1, \ldots, X_n$, then $X \to X_1 \cdots X_n$ is a ground instance of a rule from Φ;
- any other interior node (label X) has one daughter, whose label is a terminal a, such that $X \in f(a)$.

Parse trees themselves will not be the input to the learner, because this supplies too much information to make for an interesting learning problem; instead, we define a structure derived from parse trees.

Suppose then that we have an S-parse tree, and we relabel the non-leaf nodes as follows: if a node X has daughters $X_1, \ldots, X_n$, then we relabel this node with any rule schema of which $X \to X_1 \ldots X_n$ is an instance; if a node has only a terminal daughter, we relabel it with the symbol $\top$ (for "terminal"). Leaves retain their original terminal labels. Thus we now have a tree whose leaves, in order, spell out a string from the language, but the interior nodes, instead of telling us categories, only tell us which rule schema was applied at each step. A tree obtained from an S-parse tree in this manner is called a *rule structure*. (What we call "rule structures" are called "functor-argument structures" in [43], in the context specifically of classical CGs.) Thus, for example, the parse tree (17) gives us the following rule structure. We have used $/_{\mathrm{L}}$ and $/_{\mathrm{R}}$ as abbreviations for the left and right application rules, and have assumed that *and* is just lexically listed as category $(\mathrm{NP}/_{\mathrm{L}}\mathrm{NP})/_{\mathrm{R}}\mathrm{NP}$.



(33)

The set of all rule structures that can be thus generated from the grammar G is the *structure language* of G, and will be denoted $S(G)$. For distinctness, the set of strings generated by the grammar G will be called the *string language* of G and denoted $L(G)$. A rule structure gives us a unique corresponding string in the (string) language L generated by G, obtained by reading off the leaves in order. Every string in L corresponds to at least one rule structure, but possibly more than one. Thus determining the structure language of an unknown grammar certainly suffices in order to determine the string language.

Now we can see how to frame a learning problem for structures. Just as L is a subset of the countable set Σ^*, $S(G)$ is a subset of the countable set Δ of ordered trees whose leaves are labeled by elements of Σ and whose intermediate nodes are labeled by elements of $\Phi \cup \{\top\}$. Now suppose that the rule schemata of Φ are *pure*: all their terms are categories over Ω (i.e. they have no actual categories of C as subtypes). Then knowing O and Ω suffices in order for us to express the rule schemata of Φ. This means, in turn, that if we regard O, Ω, Φ, Σ as fixed, then Δ is defined. So we have a setting in which to pose a learning problem. Each possible triple (C, S, f) gives us a GCCG G, as discussed back in Section 3, and in particular we get a structure language $S(G) \subseteq \Delta$. If we consider some particular class of such triples, we get a class of structure languages, and can ask whether or not this class is learnable.

6 Learning categorial grammars

The goal of this section is to show that, under appropriate conditions, categorial grammars give us learnable classes of languages. We will study learning of structure languages and learning of string languages, and will obtain some results about both, although we are interested mainly in the latter.

So, continuing from the last section, suppose that the terminal alphabet Σ and O, Ω are fixed, and suppose that we also have a fixed set of pure rule schemata Φ (i.e. rule schemata only involving categories over Ω). For succinctness, we will henceforth use the term *categorial grammars* (CGs) to mean triples (C, S, f), and we will speak of the languages they generate, implicitly with respect to O, Ω, Φ.

It is apparent from the results we have presented so far that we cannot, in general, expect the class of all languages generated by CGs with these rule schemata to be learnable. Indeed, by Theorem 4, even the classical rule schemata give us all possible context-free languages not containing ϵ, so by a trivial variant of Theorem 10, these are not learnable from strings. In fact, we do not even need this much technology: under almost any reasonable set of CCG schemata you can think of, it is not difficult to construct a sequence of grammars $G_1, G_2, \ldots$ with proper inclusions $L(G_1) \subset L(G_2) \cdots$, and another grammar G such that $L(G_i) \subset L(G)$ for all i; once this is accomplished, Lemma 9 readily tells us that even the set $\{G_i\} \cup \{G\}$ is unlearnable. However, we can achieve learnability for the class of categorial grammars if we introduce the restriction of "rigidity." A CG $G = (C, S, f)$ is *rigid* if f assigns at most one category to each element of Σ.

As we shall see, under the assumption of rigidity, we can obtain a natural lattice structure on the class of CGs (over a fixed terminal alphabet). Grammars that are higher on the lattice generate larger languages. We can then construct an algorithm to learn a CG from a text of rule structures by successively "unifying" grammars given by the individual structures, working its way up the lattice, and eventually arriving at the correct language. Eventually we will get to talk about learning from strings as well. Our exposition will still be taken largely from Kanazawa's book [43], with adaptations to the GCCG framework and notation; much of the original work on unification in the context of categorial grammars was done by Buszkowski ([12], [14]). There is in fact a connection between this unification and the unification mentioned earlier for feature structures, but we need not concern ourselves with this.

Recall that if we had two sets of primitive categories C_1, C_2, and the category spaces over them, V_1 and V_2 respectively, then a homomorphism was a map $\phi : V_1 \to V_2$ satisfying $\phi(X|Y) = \phi(X)|\phi(Y)$. A homomorphism is uniquely determined by its values on elements of C_1. If $\phi(S)$ is again a primitive category, then for any categorial grammar G with primitive types C_1, we will write $\phi(G)$ for the grammar with primitive types C_2, obtained by applying ϕ to every element of $f(a)$ for each $a \in \Sigma$ (and using $\phi(S)$ as the start symbol instead of S). That is, $\phi(G) = (C_2, \phi(S), \hat{f})$, where $\hat{f}(a) = \{\phi(X) \mid X \in f(a)\}$. This map of categorial grammars $G \mapsto \phi(G)$ will also be called a *substitution*.

If there are substitutions ϕ, ϕ' such that $G_2 = \phi(G_1)$ and $G_1 = \phi'(G_2)$, then G_1, G_2 differ only by a relabeling of primitive categories — that is, we may assume ϕ, ϕ' are induced by bijections on the sets of primitive categories. (This is easy to see if we define the *length* of a category over C by $l(X) = 1$ for $X \in C$, and $l(X|Y) = l(X) + l(Y)$. Then applying a homomorphism can never decrease the length of a category X, and will preserve it intact only if the homomorphism sends all the primitive subtypes of X to primitive types.) It is straightforward to see that this relabeling relation is an equivalence relation on categorial grammars.

A substitution ϕ is called *faithful* (with respect to a grammar G on which it acts) if, for each $a \in \Sigma$, ϕ is one-one on the set $f(a)$. Given two grammars G, G', with respective category assignment functions f, f', we write $G \sqsubseteq G'$ if there is some faithful substitution ϕ on G such that $\{\phi(X) \mid X \in f(a)\} \subseteq f'(a)$ for each $a \in \Sigma$. That is, G' is obtained from G by applying a substitution ϕ and then possibly adding more category assignments.

Lemma 12 *$\sqsubseteq$ is a partial ordering on (equivalence classes of) categorial grammars.*

Proof: Reflexivity is immediate, as is transitivity (if $G \sqsubseteq G'$ via the substitution ϕ and $G' \sqsubseteq G''$ via the substitution ϕ', then $G \sqsubseteq G''$ by the substitution $\phi' \circ \phi$, which is still faithful). So we only have to check antisymmetry. If G, G' have respective assignment functions f, f', and $G \sqsubseteq G'$ via the faithful substitution ϕ, then $|f(a)| \leq |f'(a)|$ for each $x \in \Sigma$. And if equality holds for all a, then $G' = \phi(G)$ (with no extra type assignments thrown in). So if $G \sqsubseteq G'$ via ϕ and $G' \sqsubseteq G$ via ϕ', then we must have $|f(a)| = |f'(a)|$ for all a, hence $G' = \phi(G)$ and $G = \phi'(G')$, that is, the two grammars are equivalent. □

Now let's prove a couple nice basic properties of this ordering.

Lemma 13 *If $G \sqsubseteq G'$, then $S(G) \subseteq S(G')$ and $L(G) \subseteq L(G')$.*

Proof: Let ϕ be the substitution by which $G \sqsubseteq G'$. If $X \to X_1 \cdots X_n$ is a ground instance of a rule schema R, then for any homomorphism ϕ, $\phi(X) \to \phi(X_1) \cdots \phi(X_n)$ is again a ground instance of R. (Notice that this requires our rule schemata to be pure.) It follows that, given any derivation D over the grammar G, we can obtain a derivation D' over G' by applying ϕ to every nonterminal at every step of the derivation; the rule schema applied at each step is the same as before. Since the rule structures of $S(G)$ record only the rule schemata used and the terminals, not the actual categories that are assigned, the rule structure obtained from D' is identical to that obtained from D. Hence $D \in S(G')$, from which we have $S(G) \subseteq S(G')$. By reading off the leaf labels from each rule structure, it immediately follows that $L(G) \subseteq L(G')$. □

Lemma 14 *For any categorial grammar G', there are (up to equivalence) only finitely many grammars G such that $G \sqsubseteq G'$.*

Proof: Recall the length function l; let the *size* of a grammar G be the quantity $\sigma(G)$ defined by

$$\sigma(G) = \sum_{a \in \Sigma} \sum_{X \in f(a)} l(X).$$

Then, from $l(X) \leq l(\phi(X))$, it follows that any grammar G such that $G \sqsubseteq G'$ must satisfy $\sigma(G) \leq \sigma(G')$. Since there can only be finitely many distinct grammars (up to equivalence) of any given size, the result follows. □

Henceforth, we will restrict ourselves to *rigid* grammars. When we speak of the partial ordering $\sqsubseteq$, we will mean only the class of rigid grammars under this ordering, not all possible categorial grammars, unless otherwise indicated.

Now, to get to the point where we can demonstrate just what's so exciting about the partially ordered set of (rigid) categorial grammars, it will be necessary to introduce just one more major notion, that of *unification* of categories. Unification plays a major role in automated theorem-proving and is in fact the basis of the computer language PROLOG ([27]).

Suppose C_1, C_2 are sets of primitive categories, V_1, V_2 the corresponding category spaces over them, and $\phi : V_1 \to V_2$ a homomorphism. If $D \subseteq V_1$ is a finite set of categories, we say that ϕ is a *unifier* of D if $\phi(X) = \phi(Y)$ for all $X, Y \in D$. If $\mathcal{S}$ is a finite collection of finite subsets of V_1, then ϕ is a *unifier* of $\mathcal{S}$ if it is a unifier of every set of $\mathcal{S}$. We say that ϕ is a *most general unifier* of $\mathcal{S}$ if for every unifier $\psi : V_1 \to V_3$ of $\mathcal{S}$ (where V_3 is a category space over any C_3), we have some homomorphism $\theta : V_2 \to V_3$ such that $\psi = \theta \circ \phi$.

The relevant result for our purposes is the following standard theorem:

Theorem 15 *Let V be a category space over some set C. If $\mathcal{S}$, a finite collection of finite subsets of V, has a unifier, then it has a most general unifier.*

The proof, while not immediate, is rather unsurprising. Since discussion of unification would take us pretty far afield from categorial grammars and learning, we defer the proof to the appendix, so that we can return to our main focus. The following statement makes clear why we care about unification:

Theorem 16 *If G_1, G_2 are rigid grammars that have an upper bound under the partial ordering $\sqsubseteq$ (i.e. a rigid grammar G such that $G_1 \sqsubseteq G$ and $G_2 \sqsubseteq G$), then they have a least upper bound — an upper bound G such that any other upper bound G' satisfies $G \sqsubseteq G'$.*

Proof: We may assume without loss of generality that G_1 and G_2 use disjoint sets of primitive categories C_1, C_2. Then let $C_0 = C_1 \cup C_2$, and let V_0 be the category space over C_0. Let G_0 be the non-rigid grammar over C_0 given by assigning to each $a \in \Sigma$ the types $f_1(a) \cup f_2(a)$.

Now, suppose the rigid grammar G over a set C of primitive categories (with category space V) is an upper bound for G_1, G_2, with $G_1 \sqsubseteq G$ via ϕ_1 and $G_2 \sqsubseteq G$ via ϕ_2. Combining ϕ_1 and ϕ_2 gives us a map $\phi : C_0 \to V$, which then extends uniquely to a homomorphism $V_0 \to V$ (that agrees with the ϕ_i on V_i). Since G is rigid, ϕ must map $f_1(a)$ and $f_2(a)$ to the same category (whenever both are defined), and must also map the two start symbols S_1, S_2 both to the start symbol of G. In other words, ϕ is a unifier for the class

$$\mathcal{S} = \{f_1(a) \cup f_2(a) \mid a \in \Sigma\} \cup \{\{S_1, S_2\}\}.$$

Conversely, we can see that if ϕ is a unifier for this class, then $\phi(G_0)$ is a rigid grammar and an upper bound for G_1, G_2.

So, since G_1, G_2 have an upper bound, there exists a unifier for $\mathcal{S}$. Let ϕ now be a most general unifier, and let $G = \phi(G_0)$. We have seen that G is a rigid grammar, and an upper bound for G_1, G_2. We claim this G is the desired least upper bound. Indeed, if G' is another (rigid) upper bound, then we obtain ϕ' by the above procedure, again a unifier for $\mathcal{S}$. So we can write $\phi' = \theta \circ \phi$ for some homomorphism θ, and it immediately follows that $G \sqsubseteq G'$ via θ (it is trivial that θ is faithful on G since G is rigid). □

This allows us to prove:

Corollary 17 *The partially ordered set of rigid grammars, with a "top" element* 1 *added, forms a complete lattice: every subset has a (unique) least upper bound and greatest lower bound.*

Proof: We first show that every subset K has a least upper bound. First suppose K has an upper bound G, an actual grammar. Then, by Lemma 14, K must be finite. So let the elements of K be $G_1, G_2, \ldots, G_n$. We apply Theorem 16 to G_1, G_2 to obtain a least upper bound H_2; then apply the theorem again to H_2, G_3 to obtain a least upper bound H_3; and so forth. Eventually we get to some H_n, which is readily checked to be a least upper bound for all the elements of K. (Actually, we have one technicality: if K is the empty set, then the least upper bound is the "empty" grammar whose f assigns the empty set to every terminal.)

If on the other hand K does not have an upper bound that is a rigid grammar, then the top element 1 is the unique upper bound for K, hence it is the least upper bound.

Now we have shown that every set of grammars has a least upper bound. But if a partially ordered set has the property that every subset has a least upper bound, then every subset also has a greatest lower bound. This is a standard fact from combinatorics ([32, ch. 1]): given any subset S, let T be the set of all its lower bounds. Now if $s \in S$ and $t \in T$, then $t \sqsubseteq s$, so every element of S is an upper bound for T. Let x be the least upper bound for T (which we know exists). For any $s \in S$, the fact that s is an upper bound for T gives $x \sqsubseteq s$, hence x is itself a lower bound for S. But since $t \sqsubseteq x$ for all $t \in T$, we see that x is the greatest lower bound for S. □

This proposition is to some degree of purely academic interest; the last part of this proposition shows that sets of grammars can not only be "united" but also "intersected," but this is not an operation we presently need to use. On the other hand, we will use the upper-bound operation enough that we should establish a separate notation for it: we let $G_1 \sqcup G_2$ denote the least upper bound of G_1, G_2; or $\bigsqcup_i G_i$ denote the least upper bound of some collection of grammars G_i.

We are almost ready to put together our learning algorithm for categorial grammars. It remains only to have a way of converting the input to the learner — which, currently, consists of rule structures — into grammars.

Lemma 18 *Let T be a rule structure that can be generated by some rigid grammar G. There is a rigid grammar G with the following property: for any rigid G', $T \in S(G')$ if and only if $G \sqsubseteq G'$.*

This G will be called the *general form* associated to the rule structure T. The proof of its existence is another straightforward definition-chase, which we therefore leave for the appendix.

With this statement accepted, we can at last assemble the various formalities we have defined to obtain a learning algorithm for rigid categorial grammars.

Theorem 19 *Let Λ be the learning function on sequences of rule structures, defined as follows: If input rule structures $T_1, \ldots, T_n$ determine general forms $G_1, G_2, \ldots, G_n$, then let $G = G_1 \sqcup \cdots \sqcup G_n$. If $G = 1$ then let $\Lambda(T_1, \ldots, T_n)$ be arbitrary; otherwise let $\Lambda(T_1, \ldots, T_n) = S(G)$. Then Λ learns the class of rigid grammar structure languages.*

Proof: Let G be a rigid grammar and $T_1, T_2, \ldots$ a text for the structure language $S(G)$. So these determine general forms $G_1, G_2, \ldots$, respectively. Let $H_n = G_1 \sqcup \cdots \sqcup G_n$. Thus,

$$\Lambda(T_1, \ldots, T_n) = S(H_n).$$

Because H_{n+1} is also an upper bound of $G_1, \ldots, G_n$, we have $H_n \sqsubseteq H_{n+1}$. But since $T_i \in G$ for all i, we have that G is an upper bound for all the G_i, so $H_n \sqsubseteq G$ for each n. Hence, by Lemma 14, there are only finitely many possible values for the H_n (up to equivalence). This means that the chain

$$H_1 \sqsubseteq H_2 \sqsubseteq H_3 \sqsubseteq \cdots$$

must eventually be constant (up to equivalence) — i.e. there is some n such that $H_n = H_{n+1} = \cdots$. Let H be this constant grammar. Now by Lemma 13, $S(H) \subseteq S(G)$. On the other hand, let T be any rule structure in $S(G)$. We must have $T = T_N$ for some N, which means $T \in S(G_n) \subseteq S(H_n)$ whenever $n \geq N$, and then taking large enough n gives $H_n = H$. This shows $S(G) \subseteq S(H)$, and so we have equality. This shows that Λ has learned the structure language $S(G)$. □

Abstracting away from the particular learning algorithm, we have shown:

Corollary 20 *For a fixed terminal alphabet and set of pure rule schemata, the class of structure languages $S(G)$, where G ranges over rigid grammars, is learnable.*

□

This in itself is a rather appealing result. In fact, while we have introduced learning from structures as an auxiliary tool, it is not without merit of its own: within the assumptions of the Gold learning model, it is not far-fetched to imagine that a hypothetical language learner is able to construct rule structures for the sentences he or she receives as input (based on information about their meanings; for more on the human learner's access to semantic information, see below). The learner would then apply the above learning algorithm to learn the syntax of the language. However, we do not rest here, since we have been committed to focus primarily on string languages rather than structure languages. Hence, our next goal is to show learnability of string languages, under suitable circumstances.

As it turns out, there is some subtlety involved here. We will need to assume that the set of rule schemata Φ, in addition to being pure, has the following two properties:

- (Finite interpretability) For every string w, there are only finitely many rule structures that can be assigned to w. (That is, the set $\bigcup S(G)$, where G ranges over all rigid categorial grammars with the given rule schemata, contains only finitely many different rule structures for the terminal string w.) This is achieved, for example, if Φ has finitely many rules and the right-hand side of every rule has at least two terms — since then the rule structures are all labeled trees where each internal node is branching, and there are only finitely many such trees with $|w|$ leaves.

- (Finite elasticity) There does not exist an infinite sequence of rigid grammars
$$G_1 \sqsubseteq G_2 \sqsubseteq \cdots$$
such that the inclusions
$$S(G_1) \subset S(G_2) \subset \cdots$$
are all proper.

In fact, what we are calling "finite elasticity" is not exactly the definition used in the literature (originally from [78], [46]), but in the case of rigid categorial grammars it turns out to be equivalent.

Kanazawa demonstrates in [43] that classical categorial grammars are learnable from strings, but the proof he gives, which we shall basically follow, does not immediately generalize to a wide range of other GCCG rule systems. It is clear that the classical CG rules (8) give finite interpretability for each string (since both rules' right-hand sides have more than one term). It is less clear that the finite elasticity property holds. The day is saved by

Lemma 21 *Under the classical CG rule system (8), finite elasticity holds.*

Although the proof is nontrivial and by no means purely technical, presenting it here would interrupt the discussion of learnability, so we will again defer it until the appendix. Our proof is based loosely on that given by Kanazawa, but we make crucial use of a fairly nontrivial result, Sublemma 26, which we prove entirely independently (it plays roughly the same role in the proof as Lemma 5.56 of [43]). In the process, we develop some further theory of category spaces. We also give, in the appendix, some examples of further rule schemata one could add to Φ that would still preserve the finite elasticity property, but we will not state the results right here, since they are not sufficiently general to be worth the interruption.

Accepting Lemma 21 as true, we can go ahead and prove learnability from strings:

Theorem 22 *If the rule system Φ satisfies the conditions of finite interpretability and finite elasticity, then the class of string languages $\{L(G)\}$, as G ranges over the rigid categorial grammars, is learnable.*

Thus, in particular, the string languages generated by rigid classical CGs form a learnable family.

The idea of the proof is to use the partial-ordering apparatus we have developed for structure languages, together with Angluin's Lemma 9 for string languages. Finite interpretability suffices to make the connection between strings and structures, but we need to be somewhat careful about how this is done.

Proof: We will show that for each rigid categorial grammar G, there is a finite subset $T_G \subseteq L(G)$ with the following property: there is no grammar G' such that $T_G \subseteq L(G')$ and $L(G')$ is a proper subset of $L(G)$. The learnability result will then follow from Lemma 9. So suppose that some G does not have this property, and seek a contradiction.

Let $w_1, w_2, \ldots$ be any text of G. By assumption, for each $n \geq 1$, there exists a categorial grammar G_n with $w_1, \ldots, w_n \in L(G_n)$ and $L(G_n) \subset L(G)$. Let $S_{i,n}$ be the set of all rule structures assigned to the string w_i in $S(G_n)$. Thus, for fixed i, $S_{i,n}$ is a subset of the set of possible rule structures for w_i, which is a finite set (by finite interpretability), hence $S_{i,n}$ can take only finitely many possible values as n ranges over the positive integers.

Now we show that there exist sets $S_1, S_2, \ldots$ with the following property: for each k, there exist infinitely many n such that the equations

$$S_{1,n} = S_1, \quad S_{2,n} = S_2, \quad \ldots, \quad S_{k,n} = S_k$$

simultaneously hold. This holds by an easy induction on n. For $k = 1$ it holds because the $S_{1,n}$ have only finitely many possible values, so infinitely many of them must be equal, and we take S_1 to be this common value. Similarly, if the claim is proven for $k-1$, then we look at all the values of n such that

$$S_{1,n} = S_1, \quad \ldots, \quad S_{k-1,n} = S_{k-1}.$$

There are infinitely many such n, so among the corresponding values of $S_{k,n}$, some value must occur infinitely often, and we take S_k to be this value; thus we get the induction step.

Now, for each i, let G'_i be the rigid grammar that is the least upper bound of the general forms for the rule structures in S_i. Let $H_i = G'_1 \sqcup G'_2 \sqcup \cdots \sqcup G'_i$. Each H_i must be an actual grammar (not the top element), since there are infinitely many grammars G_n that generate all the rule structures in $S_1, \ldots, S_i$, and therefore are $\sqsupseteq H_i$. We have the sequence

$$H_1 \sqsubseteq H_2 \sqsubseteq H_3 \sqsubseteq \cdots.$$

Therefore, the chain of structure languages

$$S(H_1) \subseteq S(H_2) \subseteq S(H_3) \subseteq \cdots$$

must eventually be constant: indeed, if not, then there are infinitely many j for which $S(H_{j-1}) \subset S(H_j)$, and taking those H_j would give a contradiction to finite elasticity.

Thus, there is some N such that $S(H_i) = S(H_N)$ for all $i \geq N$. Since $S_i \subseteq S(G'_i) \subseteq S(H_i)$, we have $S_i \subseteq S(H_N)$ for all i. In particular, H_N generates the string w_i — that is, $w_i \in L(H_N)$ for all i. Thus,

$$L(G) \subseteq L(H_N). \tag{34}$$

On the other hand, there is some n (indeed, infinitely many) such that $S_{1,n} = S_1, \ldots, S_{N,n} = S_N$. Then, since G_n generates all the structures in $S_1, \ldots, S_N$, we must have $H_N \sqsubseteq G_n$ and therefore (by Lemma 13)

$$L(H_N) \subseteq L(G_n) \subset L(G) \tag{35}$$

where the second inclusion is strict, by assumption. This, and (34), give our needed contradiction. □

So, we have successfully proven learnability of at least some rigid grammars (such as classical CGs) from strings, using finite interpretability and finite elasticity. Notably, in our application of Lemma 9, we did not actually construct tell-tale subsets for the language $L(G)$; we only have an existence proof.

As Kanazawa notes, this proof is not adequate for effective learning (i.e. learning by a computable function). The problem is that the proof of Lemma 9 requires computing a grammar G such that $L(G)$ is minimal with the property that $w_1, \ldots, w_n \in L(G)$. Unfortunately, we don't know how to determine minimality; the problem of determining whether $L(G_1) \subseteq L(G_2)$ is in fact undecidable at least for general categorial grammars (because the corresponding problem for context-free grammars is undecidable, [33, ch. 8]). However, Kanazawa gives a computable learning function based on the idea of determining minimality "in the limit."

The idea is as follows: Given strings $w_1, \ldots, w_n$, there is a finite set of grammars $W_n = \{G'_1, \ldots, G'_m\}$, each of which can generate all of these strings, such that if G' is any other language that generates all these strings, then $G'_j \sqsubseteq G'$ for some j. (Proof: consider all n-tuples $(S_1, \ldots, S_n)$ such that S_i is a rule structure for w_i; there are finitely many such n-tuples. For the j-th n-tuple, let G'_j be the least upper bound of the general forms given by the rule structures in that n-tuple.)

Now let $G_1, G_2, \ldots$ be a (computable) enumeration of all possible categorial grammars; and let $v_1, v_2, \ldots$ be a (computable) enumeration of all possible strings. Let W'_n be the set of all $G' \in W_n$ for which $L(G') \cap \{v_1, \ldots, v_n\}$ is minimal (among grammars in W_n). We define the learner Λ as follows: $\Lambda(w_1, \ldots, w_n)$ is the element of W'_n that occurs earliest in our enumeration of all possible grammars. This is effectively computable: we can determine $L(G') \cap \{v_1, \ldots, v_n\}$ since we can determine for any particular $v \in \Sigma^*$ whether or not $v \in L(G')$ (just compute all the possible rule structures for it and see if any of them gives a general form that is $\sqsubseteq G'$).

We need to show that this Λ successfully learns rigid categorial grammars from strings. Well, let G be a grammar for the language to be learned, and also fix a text $w_1, w_2, \ldots$. For each i, let T_i be a rule structure assigned by G to w_i (if there is more than one such structure, pick one), and let H_i be the grammar produced by running our earlier structure-learning algorithm on the input $T_1, \ldots, T_i$. Then $H_i \sqsubseteq G$ for all i, and

$$H_1 \sqsubseteq H_2 \sqsubseteq \cdots,$$

so by Lemma 14, the sequence of H_i must eventually be constant. Let H be its value. Then $L(G) \subseteq L(H)$ since H generates all of the strings $w_1, w_2, \ldots$; but also $H \sqsubseteq G$ implies $L(G) \subseteq L(H)$, so we have equality.

By construction of H, we see that H is in the set W_n defined above when n is sufficiently large. We claim that, in fact, for sufficiently large n, H is in fact in W'_n. Indeed, we know by the proof of Theorem 22 that there exists some $T_H \subseteq L(H)$ such that, for any rigid grammar H' with $T_H \subseteq L(H')$, $L(H') \not\subset L(H)$. Suppose N is sufficiently large so that $T_H \subseteq \{w_1, \ldots, w_N\}$. Now, by construction of the sets W_n, we have the following: if $H'_n \in W_n$ for $n \geq N$, then there is some $H'_N \in W_N$ with $H'_N \sqsubseteq H'_n$. (Specifically, H'_n is obtained by unifying the general forms of some rule structures for $w_1, \ldots, w_n$; just unify the first N of these to get H'_N.) But for every $H'_N \in W_N$, we have

$$T_H \subseteq \{w_1, \ldots, w_N\} \subseteq L(H'_N)$$

which means that $L(H'_N)$ contains some string $w_{H'_N} \notin L(H)$ (unless it happens that $L(H'_N) = L(H)$). When n is sufficiently large, then we have

$$w_{H'_N} \in \{w_1, \ldots, w_n\} \cap \{v_1, \ldots, v_n\}$$

for every choice of H'_N. At this point, then, for any $H'_n \in W_n$ such that $H \not\sqsubseteq H'_n$, we have some choice of $H'_N \sqsubseteq H'_n$ so that $w_{H'_N} \in L(H'_N) \subseteq L(H'_n)$ and hence

$$L(H'_n) \cap \{v_1, \ldots, v_n\} \not\subseteq L(H) \cap \{v_1, \ldots, v_n\}.$$

This shows that H meets the minimality condition to be in W'_n.

So H, which generates the same string language as G, is in W'_n for all sufficiently large n. H appears somewhere in our enumeration $G_1, G_2, \ldots$ of grammars, say $H = G_k$. Now, we have defined Λ to pick out a member of W'_n that occurs as early as possible in the enumeration, hence $\Lambda(w_1, \ldots, w_n) = G_j$ for some $j \leq k$. We now need only show that each G_j for $j < k$ will be chosen at most finitely many times, except possibly if $L(G_j) = L(H)$. Then it will follow that the learner will eventually settle either on G_k or on some earlier G_j that generates the same string language, and this will prove our result.

Consider any $j < k$. If $L(G_k) \subset L(G_j)$ (properly), then some v_N is in $L(G_j)$ but not $L(G_k)$. Then minimality will be violated — that is, $G_j \notin W'_n$ for $n \geq N$, and so the learner will never again output G_j. If on the other hand $L(G_k) \not\subseteq L(G_j)$, then there is some w_N in $L(G_k)$ but not $L(G_j)$, and so $\Lambda(w_1, \ldots, w_n) \neq G_j$ when $n \geq N$ (since, by construction, our learner only guesses grammars that are consistent with the data so far). So every G_j for $j < k$, such that $L(G_j) \neq L(G_k)$, is eventually eliminated, and we thus have a *computable* learner of the string languages of rigid categorial grammars.

Now, from all the notational mess, this learning algorithm seems a bit too complicated and computation-intensive to be a realistic model of the human learning process, so we should pause a moment in our formal study to discuss what is really going on. The idea of Lemma 9 — which is arguably the main ingredient in proving learnability of categorial grammars from strings, Theorem 22 — is that we construct a learner which hypothesizes the smallest possible language consistent with the data given so far. All of the remaining formal baggage involved in the proof of Theorem 22 is then used in proving that this learner actually learns the target language (as opposed to, say, guessing successively larger and larger languages but never reaching the target); the learner itself does not make use of any of this formalism.

Now, the foregoing discourse focused on the fact that, in practice, “the smallest possible language consistent with the data given so far” may not be computable. (Alternatively, it may be computable only in an unreasonable amount of time.) The conceptual solution to this should be to use some method of approximating the smallest possible language that computes “in the limit” what this smallest language is. The computable learner we gave above does this in perhaps the most crude and naive way possible, finding the minimal language in the limit by finding the language whose intersection with $\{v_1, \ldots, v_n\}$ is minimal, for increasingly large values of n. Hence, it seems that one place to work on the model to make the learner more realistic would be to find more efficient ways of approximating the smallest language, at least for specific classes of categorial grammars. For now, we simply point to this goal as one for future research, and move on to discuss other cases of learnability.

We have so far focused on rigid grammars; although this makes for a relatively nice theory, the constraint is, well, rigid. An immediate way to relax the requirement of rigidity, proposed by Kanazawa in [43], is to consider the *k-valued* grammars: these are categorial grammars for which the function f assigns at most k categories to each terminal. There is a very simple relation between k-valued grammars and rigid grammars. Let Σ' be the disjoint union of k copies of Σ, and let $\phi : \Sigma' \to \Sigma$ be the map that sends each copy in Σ' of any element $a \in \Sigma$ to a. This extends in an obvious way to a homomorphism $\phi : \Sigma'^* \to \Sigma^*$. We can define $\phi(L)$ for a language L over Σ' (as the set $\{\phi(w) \mid w \in L\}$). If we also define $\phi(T)$, for any rule structure T, to be the rule structure obtained by applying ϕ to each terminal label on a leaf of T, then we can look at the image $\phi(S)$ where S is a structure language over Σ', thus getting a structure language over Σ.

Lemma 23 *The string languages generated by k-valued grammars over Σ are precisely those that can be expressed in the form $\phi(L(G'))$, where G' is a rigid grammar over Σ'. The structure languages generated by k-valued grammars over Σ are precisely those that can be expressed in the form $\phi(S(G'))$, where G' is a rigid grammar over Σ'.*

Proof: It suffices to prove the second statement, as the first follows from it. If $G' = (C, S, f)$ is a rigid grammar over Σ', then define a k-valued grammar $(C, S, \hat{f})$ over Σ by setting $\hat{f}(a) = \cup_{\phi(b)=a} f(b)$ — that is, $\hat{f}$ assigns to each terminal all of the types assigned to its various copies in Σ' by f. It is straightforward to check that the structure language generated by this grammar is $\phi(S(G'))$. Conversely, given a k-valued grammar $G = (C, S, \hat{f})$ over Σ, define a rigid grammar $G' = (C, S, f)$ over Σ' as follows: if $\hat{f}$ assigns j ($\leq k$) categories to a terminal $a \in \Sigma$, then f should assign these categories to j copies of a in Σ' (and assign no categories to the remaining copies). Again, it is immediate that $S(G) = \phi(S(G'))$. □

As Kanazawa shows, we can exploit this connection to prove learnability of the languages generated by k-valued grammars, under suitable conditions:

Theorem 24 *If finite elasticity holds, then for each $k \geq 1$, the class of structure languages $\{S(G)\}$, where G ranges over the k-valued grammars, is learnable; if we have finite interpretability as well, then the class of string languages $\{L(G)\}$ is also learnable.*

Rather than go through the tedium of laying out the full proof here, we will just give the basic idea. It suffices to construct a learner that computes in the limit a rigid grammar G' over Σ' satisfying $\phi(S(G')) = S(G)$ (or $\phi(L(G')) = L(G)$, respectively); by Lemma 23, such a grammar does exist. Consider the case of structures (the proof for strings is entirely analogous). For each rule structure T over Σ, there are finitely many rule structures T' over Σ' such that $\phi(T') = T$. Hence we can simply duplicate the proof of Theorem 22, with structures over Σ playing the role occupied by strings in the previous proof, and with structures over Σ' playing the role previously played by structures over Σ. Finite elasticity is now applied to grammars over Σ'.

Really, the proofs of Theorems 22 and 24 are the same, at a suitably abstract level. Both involve the reduction of one learning problem to another: in both cases we have a map between some two infinite sets, $\phi : \Delta' \to \Delta$, and grammars G generating subsets $D(G) \subseteq \Delta'$. (In the former case, ϕ is the map from rule structures to strings that reads off terminal labels; in the latter, it is the map $\Sigma'^* \to \Sigma^*$ that we just defined). As long as $\phi^{-1}(x)$ is finite for every $x \in \Delta$, we can use the idea of finite elasticity to infer the learnability of the class $\{\phi(D(G))\}$ from the learnability of the class $\{D(G)\}$. (This general method of proof is analogous to Theorem 3.1 from [43], but our use of it is somewhat more succinct.)

Unfortunately, for computational purposes, the resulting algorithm for learning k-valued structure languages suffers the same excessive complexity that we saw with the learning of rigid string languages (and of course this is all the more so if we want to learn k-valued string languages). The same comments made before apply again here: the difficulty is in computing, or approximating, the *minimal* k-valued language that contains a given set of structures (or strings), but one can at least prove that such a minimal language exists. Work on simpler, faster algorithms that compute the minimal language in the limit would be of interest.

In closing up our discussion of learning categorial grammars, we should think seriously about what more appropriate restrictions we could impose, instead of rigidity. To do so, we should be guided by trying to consider the ways in which actual human language violates rigidity. It is important to keep in mind what the elements of Σ are to represent, a point on which the whole literature is somewhat vague. If they represent, say, phonological forms of words, then rigidity is extremely problematic, because clearly a single word can have a host of categories (think of English *set*, which is a noun, a verb, and an adjective, not to mention its many possible subcategorizations). We prefer to suppose that elements of Σ are abstract lexical items, with each possible meaning assigned to a different terminal symbol, and different terminals sometimes having the same phonological form. (We might even posit a single universal vocabulary Σ of possible meanings for all human languages, with the choice of how to bundle these meanings together into units with a single phonological form left up to each individual language.)

What this entails for our theory of learnability, then, is the assumption that the learner has access not only to the string of phonemes but also to the particular lexical item that each phonological word references. This might seem like a concession since we have until now been concerned exclusively with syntactic information. But from the point of view of modeling human language acquisition, it is no problem at all. Obviously any theory of language acquisition will ultimately need to explain semantic as well as syntactic acquisition, which means it will need to posit that learners have information about the meanings of sentences they receive as input; in view of this, accommodating semantic information by assigning different terminal symbols to different meanings is hardly a shortcoming of our learning framework. Similarly, as hinted earlier, the idea that rule structures themselves would be available to the learner is not entirely ridiculous if the learner knows what different semantic operations correspond to each rule schema.

However, there is another respect in which rigidity poses a more serious problem, and this is with regard to the idea of features. As described at the end of Section 3, there is no especially neat way to incorporate features into our CG framework; the only solution is to take what we normally think of as a primitive category (such as N or S) and subdivide it into many categories based on all possible feature bundles that can be attached to it, then give each lexical item a pile of different categories, one for each valid assignment of feature bundles to its primitive subtypes. This results in an extremely non-rigid grammar system. Worse, if there are, say, c feature structures that can be associated with each primitive category, then a "featureless" category of length l would split into c^l possible categories with features, and this number is unbounded if we do not a priori have a bound on l. Thus we no longer are even assured of falling within the k-valued class of grammars for any particular k.

Really, though, there seems to be a more fundamental problem with the CCG handling of features vis-à-vis our model of learning. Clearly, when speakers of a language learn a new word, it does not need to be learned separately with each of its possible feature instantiations. For example, if I utter the sentence *you snarple the tree*, thus informing you that *snarple* has category $(\mathrm{S}/_{\mathrm{L}}\mathrm{NP}_{[2sg]})/_{\mathrm{R}}\mathrm{NP}$, you will also be able to use it as an $(\mathrm{S}/_{\mathrm{L}}\mathrm{NP}_{[1pl]})/_{\mathrm{R}}\mathrm{NP}$ and so forth, forming sentences such as *we snarple the pigs* without needing to learn these forms separately. Our current model of features in CCG, wherein each of these is a distinct category, has no way of capturing this phenomenon. (Actually this example is easy to handle; we could postulate that the verb has only a single category, and that there are agreement suffixes, some of which are phonologically null, that have categories for all of the different possible agreement features. This requires the language learner to learn each suffix just once in order to get the agreement features down pat, and then to learn each verb root only once. But this solution does not immediately handle other cases of feature-bundle proliferation, such as control verbs and other lexical inheritors.)

It seems that there should be a device by which the learner can "generalize" or "analogize" to conclude that, when a lexical item has certain categories, it also has certain others. However, this goes well beyond the learning framework we have discussed here; our theory, by requiring the use of general forms and operating on equivalence classes of grammars, seems to prohibit anything that would require reference to a specific choice of category. We will leave this as one more direction for future research in formal modeling of language learning.

7 Conclusion

Now that we have established some of the basic formal properties of categorial grammars, we will close by discussing the relevance of the results we have presented, as well as the relevance of our techniques more generally. As for the latter, the views we present here (unlike the assumptions described in Section 1) are the author's and are not necessarily the consensus of mathematical linguists or of the generative grammar community.

Now then, let's review the results on formal complexity of categorial grammars. Theorems 4 and 5 show that classical categorial grammars generate precisely the context-free languages (not containing ϵ); and a broader class of CG systems, namely those using only finite, depth-preserving rule systems, also generate only context-free languages. Argument-depth-preserving rule schemata give us at most linear indexed languages (our Theorem 7), providing a straightforward formal generalization of the result of [73]. We have also seen that fairly standard CCG schemata, or approximations thereto, can generate non-linear-indexed languages. These results can be interpreted as saying that certain types of CCG rule schemata systematically undergenerate, while others overgenerate (assuming our views on the complexity of $\mathcal{L}^*$ to be correct). From the point of view either of comparing the validity of different theories or of seeking evolutionary explanations, then, these results are instructive, although also limiting (for example, it seems difficult to distinguish between the many possible argument-depth-preserving rule systems on the basis of generative power).

In any event, the relevance of the tools we have presented here to understanding natural language should not be exaggerated. The basic and fairly obvious shortcoming of the whole theory is that there is no particular reason for natural language to respect the Chomsky hierarchy. The classes of regular languages, context-free languages, and so forth continue to be the standard yardstick used by linguists who want to describe the expressive power of one or another theory, and there is even an unfortunate tendency to worship them as being *the* canonical measures of complexity; thus one sees grammar systems described as "trans-context-free" ([69]) or empirical data that are "provably context-sensitive" ([77, ch. 4]). Of course they are not the only possible measures of complexity. For example, the language $\{a^{2^n} \mid n \geq 0\}$ is extremely simple, as measured naively by the number of symbols it takes to describe it, and the membership problem can be decided very efficiently, but it falls at the context-sensitive level of the hierarchy. Realistically, the machinery underlying natural language should be sensitive to complexity constraints very different from those defining the levels of the Chomsky hierarchy. These levels have direct relevance to natural language insofar as it could be believed that, say, context-free grammars are the basis of natural language; but we now know that they are not. Therefore, we should expect that the class $\mathcal{L}^*$ cross-cuts the Chomsky hierarchy, with some languages perhaps located at the context-free level and others not, while $\mathcal{L}^*$ cuts out only a thin slice of each level. (This general observation has also been made in [58] and in [45, ch. 9].)

If this is the case, why does the Chomsky hierarchy continue to play such a central role in studies of language complexity? The basic answer is that it is the best scale that we have. The simple definitions of the language classes, the fact that many non-linguistic systems mesh well with them, and the relatively clean formal properties of these language classes make them a convenient choice to work with. It is not at all clear what kinds of complexity measurements would be better-suited for constructing a system in which human languages had a natural place, aside from considerations of time and space efficiency of parsing and perhaps degrees of ambiguity (i.e. assigning multiple structures to the same string).

The most concrete task that can be accomplished by applying formal language theory to natural language is to rule out possible theories of universal grammar on grounds of undergeneration. Thus, for example, it is now safe to say that classical categorial grammar is not an adequate description of natural language, because Theorem 4 tells us that this system only generates context-free languages, whereas human languages are not context-free. There are several reasons that this is the most directly productive direction in which to apply the theory. One is that the formal theories of language that we can devise are only approximations, and it is easier to construct approximations $\mathcal{L}$ for which $\mathcal{L} \supseteq \mathcal{L}^*$ than for which $\mathcal{L} \subseteq \mathcal{L}^*$. For example: classical CG by itself is obviously too simple to give a reasonable characterization of the class of natural languages (this is a matter of common sense, not a theorem that needs proving), but the linguist who wishes to construct a more comprehensive theory of universal grammar based on classical CG might do so by adding restrictions (specifying the number of primitive categories, or limiting the lengths of categories that can be assigned to a terminal). Any such theory would then generate some subset of the class of languages generated by "bare" classical CG theory. Consequently, using Theorem 4, any theory along these lines is going to be inadequate.

Now, there are other ways of embellishing a theory of universal grammar that do not simply give a subset of the original language set; for example, we might impose a bound on the number of times a rule can be used in a derivation. But in general it is difficult to say what new classes of languages would be generated by such restrictions. Put another way: suppose that you begin with some given rudimentary theory of universal grammar (such as classical CG), which you hope to develop into a more fleshed-out, plausible theory. Suppose that the two theories are to generate language classes $\mathcal{L}, \mathcal{L}'$, respectively. If you want to carry out the development in such a way that $\mathcal{L}$ and $\mathcal{L}'$ bear any discernible relationship to each other, that relationship is likely to be $\mathcal{L}' \subseteq \mathcal{L}$. Therefore, the best way to use tools of formal language theory to try to obtain meaningful information about this hypothetical fleshed-out theory is to show that $\mathcal{L}$ is too low on the complexity hierarchy; this then guarantees that no fleshed-out theory derived in this manner can be adequate.

This, then, tells us one reason why it is more useful to prove that a theory of grammar undergenerates than it is to prove that it overgenerates: if it overgenerates, perhaps we just need to throw in more restrictions to develop the theory. Another reason comes from the empirical rather than the theoretical side: lower bounds on the complexity of natural language are more easily obtainable than upper bounds. For example, we can collect data to convince ourselves that Dutch is not context-free. But using data to show conclusively that any language *is* in some complexity class, say the linear indexed languages, is well beyond the present means of the science of linguistics — let alone showing that all human languages fall into the linear indexed class. Indeed, we probably would not be able to do that until we had an accepted comprehensive grammar for the language; as far as this author is aware, there is no technique for showing that a language falls into one of the standard complexity levels without constructing, at least implicitly, a grammar of the required sort. So we cannot "know" that a theory that gives, say, all context-sensitive languages overgenerates; we can only take it as an article of faith. (Of course, such knowledge is also not clearly meaningful because the notion of a "grammar" is a theoretical construct outside of the physical world — to say nothing of the notion of "possible human languages.") Even the available tools for showing that natural languages fall outside of some complexity class are very limited; in practice, it basically comes down entirely to the pumping lemma, or its analogues for other classes of languages. The analysis of formal complexity of languages would be greatly aided by having some other useful means of demonstrating that a language was outside a particular class.

Despite these limitations on detecting overgeneration, it can still be helpful in directing further research. Even though it may be clear that some theory is too simple to be a realistic model of universal grammar, knowing that it overgenerates helps in deciding how to improve it: there is some particular language that the revised theory will have to exclude. Or if some more fleshed-out theory is too complicated to analyze thoroughly in terms of generative power, one can at least look at any results about the simple theory on which it is based in order to decide where to look for possible examples of overgeneration.

Results that neither show overgeneration nor undergeneration, but rather suggest that a theory has the appropriate level of complexity, also have their role to play. For example, Theorem 6 does not tell us that combinatory CGs generate all linear indexed languages, nor is it entirely clear whether or not they should. But having statements like this is nonetheless useful in the quest to judge the reasonableness of the model. If one is asking "can I show that this theory overgenerates by giving a non-linear-indexed language it can produce?" then proving containment within the linear-indexed class will stop the fruitless search for counterexamples. And while no one can formally "prove" that a theory of universal grammar generates the right class of languages, if we have been unable to prove that it cannot generate the right class, then the theory may be worth further research, until someone discovers it is wrong for another reason.

In general, syntacticians try to hone in on appropriate models of grammar by studying overgeneration and undergeneration: a grammar is bad insofar as it fails to generate some sentence that should exist, or it generates some sentence that should be ungrammatical. This empirical approach is perfectly appropriate for describing individual languages, but in order to study theories of universal grammar and the undergeneration or overgeneration of *classes* of languages, we basically need formal tools of the sort used here: we need to be able to talk about structural properties of entire languages in the same way that we are accustomed to talking about structural properties of sentences. We are only beginning to develop the kind of rigor needed for this, and we often see it entirely lacking in the literature. Pullum and Gazdar's paper [55] cites linguistics textbooks asserting, for example, that "we can be sure" that natural languages cannot be described by context-free grammars, giving as reasons only the observations that the most obvious attempts to construct such grammars fail; in some cases, Pullum and Gazdar specifically show that the data used as evidence in such flimsy arguments actually can be accounted for by context-free grammars, albeit not very intuitive ones. Formal language theory gives us at least the beginnings of the machinery for making sense of, and even proving, statements about the insufficiency of a grammar system. In the future, it may give us solid means of proving the overabundance of a grammar system as well.

Now let us move on to the results about learnability theory. We saw that even the classical categorial grammars — and, indeed, virtually any "reasonable" CG system — resulted in an unlearnable class, under the Gold framework. Then we looked at what happens when we impose the restriction of rigidity. As we have seen from Corollary 20, learnability from structures does hold, for any collection of pure rule schemata, and Theorem 22 shows that this transfers to learnability of string languages in suitable cases, including for classical CGs. We have also seen that these results are (at least sometimes) applicable to k-valued grammars as well. Learnability from structures is somewhat encouraging for the framework of (rigid) categorial grammars in general, although the results we have cited give no concrete way of preferring one rule system to another. If some particular systems were known to be learnable from strings, this would give a way to prefer them; but the only such systems currently established are depth-preserving, so we have to rule them out on account of generative power. The most elegant, and perhaps the most interesting, part of this theory is not the results themselves but rather the apparatus of unification and the structure it imposes on the collection of categorial grammars, which provides a clean way to understand the learning algorithm as successively gluing together the pieces of information it receives into a rigid grammar that is as small as possible (under the partial ordering $\sqsubseteq$).

Unfortunately, when it comes to the linguistic relevance of these results from learning theory, we are on shakier ground than in the discussion of generative power. It is essentially impossible to either uphold or reject a theory of universal grammar on grounds of learnability, because this whole framework depends on an extra dose of assumptions that are approximations at best — namely, the learning model. The Gold model that we have used is an especially simple one, but any model powerful enough to make clear and interesting predictions about both learnable and unlearnable languages would have to suffer major drawbacks of relevance. Under any model now available, learnability is too weak a formal test for the validity of a theory of universal grammar, in that it fails to constrain the choice of learning algorithm so as to make it realistic. Learnability is also too strong a formal test, in that it operates on the unrealistic assumption that the child successfully learns the language of its environment, neglecting diachronic change. Our lack of certainty about the format of the input available to the learner adds to the confusion.

In other words, when we consider all of the confounding factors, it is not really clear whether, in any given model, the actual class $\mathcal{L}^*$ is learnable. And if we do not know whether $\mathcal{L}^*$ is learnable, then even when we have results about the learnability of $\mathcal{L}$ in a particular theory, we do not know whether this means we should accept or reject the theory. The situation is somewhat like accepting or rejecting a theory of extraterrestrials based on its predictions about the clothes they wear: the empirical evidence isn't really there.

Nonetheless, learnability and unlearnability are useful if one treats them only as a heuristic; a good theory should predict a language class $\mathcal{L}$ of an appropriate size, and learnability is intuitively a sign of better size than unlearnability, insofar as it means the class is manageable. Results showing that a class is learnable do not show that the theory is correct, but at least they give us reason to continue pursuing it.

Even taking into account the inadequacies of the learning model, one can get results from the application of learning theory that are valuable in other ways. We should be cautious about claiming that any particular learning algorithm is a realistic model of how human acquisition takes place; still, we can at least draw some ideas from learnability theory to formulate hypotheses about actual learning, which can then be used to further our experimental understanding of language acquisition. And, as already noted, learning algorithms do have their place in artificial intelligence, where one generally works in a paradigm that is much better-understood and more modelable than the human brain. Learnability theory also provides a formal tool whose importance was mentioned above: a way of measuring the size of a class of languages, separately from the Chomsky hierarchy. Even if it is an extremely coarse measurement, any criterion helps, when it comes to comparing classes of languages generated by various theories.

So, with the caveats we have just given, are all of these sixty pages a waste of time? No. The tools of mathematical linguistics have their proper place in helping to direct the formulation of reasonable models of universal grammar. The limitations listed just now indicate the major conceptual and methodological challenges in this field in the future. Attacking these challenges, as well as the more specific short-term problems we have described in Sections 4 and 6 about classifying categorial grammars in order to obtain more general formal results, will provide potentially very fruitful (if rather open-ended) lines for further study.

Discussing the topic of generative power, Chomsky writes in [16], "It is important to realize that the questions presently being studied are primarily determined by feasibility of mathematical study, and it is important not to confuse this with the question of empirical significance." This is true, but it is tautological; nobody studies questions whose study is infeasible. The fact that new results are driven by attainability is an inevitable property of research in any area. Although we seem to be progressing in only the minutest steps toward the goal of formulating a complete formal theory of the language faculty, and a huge amount of uncharted territory remains ahead, we can still hope that the tools we have exhibited and the applications to which we have put them will contribute toward eventual understanding.

8 Acknowledgments

The author would like to thank his advisors, Cedric Boeckx and Cliff Taubes of Harvard University, for all of their help in discussing preparations for this work and reading drafts. Pauline Jacobson's class on categorial grammar at Harvard in the fall of 2004 also offered extremely helpful exposure to the theory of categorial grammar and its underlying ideas. Michael Gensheimer and Peter Green looked at earlier drafts of this thesis and offered useful comments. Barry Mazur, Martin Nowak, Ed Stabler, and Mark Steedman have also provided helpful conversations, while Jennifer Finley and Javier Martin-Gonzalez have provided administrative help. And the crocodile did its part by eating that stale, tasteless old cake.

Any errors are of course the sole responsibility of the author. Oops, my bad.

A Proofs

Here we will include the proofs of some of the auxiliary results that were omitted from the main body of the text. First, we prove the statement that every ϵ-free context-free language is generated by a classical CG.

Proof of Lemma 8: ([9]) Suppose (V, S, P) is a grammar for the context-free language L, $\epsilon \notin L$. First suppose the grammar is in *Chomsky normal form* ([33, ch. 4]), that is, each rule is of the form

$$X \to YZ \ (X, Y, Z \in V) \qquad \text{or} \qquad X \to a \ (X \in V, a \in \Sigma).$$

Enumerate the rules of the former type as

$$X_1 \to Y_1 Z_1, \quad X_2 \to Y_2 Z_2, \quad \ldots, \quad X_m \to Y_m Z_m.$$

(These symbols are of course not necessarily distinct.) Let Q be the set of all triples (i, j, k), $1 \leq i, j, k \leq m$, such that $X_i \overset{*}{\Rightarrow} wX_j$ for some $w \in V^*$ and $Z_j = X_k$. We will construct a classical CG whose set of primitive categories C consists of the symbols in V, the numbers $1, \ldots, m$, and the triples in Q.

The category assignment function f is built as follows. We first construct a function $\hat{f}$ assigning nonterminals in V to categories; then f assigns a to category U if and only if there is some $X \in V$ for which $X \to a$ in P and $\hat{f}$ assigns X to category U. The construction of $\hat{f}$ is according to the following rules:

1. $\hat{f}$ assigns each $X \in V$ to itself;
2. $\hat{f}$ assigns each Y_i to category i;
3. if $(i, i, k) \in Q$ then Y_k gets category $(i, i, k)/_{\mathrm{L}}i$;
4. if $(i, j, k), (i, k, l) \in Q$, then Y_l gets category $(i, k, l)/_{\mathrm{L}}(i, j, k)$;
5. if X_i is assigned a category U by rules 1-4, then Z_i gets category $U/_{\mathrm{L}}i$;
6. if $(i, j, k) \in Q$ and X_i is assigned a category U by rules 1-4, then Z_k gets category $U/_{\mathrm{L}}(i, j, k)$.

The categories assigned by rules 1-4 will be called *left* categories, and the remaining ones we call *right* categories; note that these sets of categories are disjoint.

Now consider any parse tree generated by the context-free grammar. Each nonterminal that appears in the tree, except for the root, emerges as the result of a rule of the form $X \to YZ$. Each appearance of a symbol that is obtained as the Z from such a rule (i.e. each symbol that is the right daughter of its mother) will be called *right*, and the other symbols (a symbol that is the left daughter of its mother, or the root) are called *left*.

Now suppose T is a parse tree for a derivation of $U \overset{*}{\Rightarrow} U_1 \cdots U_n$ in our context-free grammar, where U and all the U_i are nonterminals. We claim that for each U_i, there is a category A_i assigned to it by $\hat{f}$ such that $U \overset{*}{\Rightarrow} A_1 \cdots A_n$ under the classical CG rules; moreover, A_j is a left category if and only if U_j is left in the tree. The proof is by induction on n; the base case $n = 1$ is obvious. Otherwise, choose an internal (non-leaf) node occurring left in the tree, such that the substring of $U_1 \cdots U_n$ that it dominates is as short as possible. Let this node be labeled W, and suppose it dominates the substring $U_p \cdots U_q$. By minimality, no internal node below W can be left, so the subderivation $W \overset{*}{\Rightarrow} U_p \cdots U_q$ must have the form

$$W \Rightarrow U_p W_p \Rightarrow U_p U_{p+1} W_{p+1} \Rightarrow \cdots \Rightarrow U_p U_{p+1} \cdots U_{q-2} W_{q-2} \Rightarrow U_p U_{p+1} \cdots U_{q-1} U_q.$$

Now apply the induction hypothesis to the tree T with the subtree below W excised; this is a parse tree for $U \overset{*}{\Rightarrow} U_1 \cdots U_{p-1} W U_{q+1} \cdots U_n$. Thus we obtain categories A_l for U_l when $l < p$ or $l > q$, and also a left category B for W. But the subderivation starting from W consists of applying rules of the form $W_{l-1} \to U_l W_l$ (where $W_{p-1} = W$ and $W_{q-1} = U_q$); hence for each such $l = p, \ldots, q-1$, we have $W_{l-1} = X_{i_l}, U_l = Y_{i_l}, W_l = Z_{i_l}$ for some integer i_l. Then

$$X_{i_p} = W_{p-1} \overset{*}{\Rightarrow} U_{p-1} \cdots U_l W_l = U_{p-1} \cdots U_{l-1} X_{i_l}$$

and $Z_{i_l} = X_{i_{l+1}}$ implies that the triples

$$(i_p, i_p, i_{p+1}), \quad (i_p, i_{p+1}, i_{p+2}), \quad (i_p, i_{p+2}, i_{p+3}), \quad \ldots, \quad (i_p, i_{q-2}, i_{q-1})$$

are all in Q. Therefore, $\hat{f}$ assigns

- the left category $A_p = i_p$ to U_p,
- the left category $A_{p+1} = (i_p, i_p, i_{p+1})/_{\mathrm{L}} i_p$ to U_{p+1},
- the left category $A_l = (i_p, i_{l-1}, i_l)/_{\mathrm{L}}(i_p, i_{l-2}, i_{l-1})$ to $U_l = Y_{i_l}$ for $p + 1 < l < q$, and
- the right category $B/_{\mathrm{L}}(i_p, i_{q-2}, i_{q-1})$ to U_q.

This category assignment meets our needs, and our claim is proven.

Now we also need to prove the converse: that if $U_1, \ldots, U_n \in V$ have categories $A_1, \ldots, A_n$, and $A \in V$ such that $A \overset{*}{\Rightarrow} A_1 \cdots A_n$ under the CG rules, then $A \overset{*}{\Rightarrow} U_1 \cdots U_n$ under the original context-free rules. We first make a simple observation: if A, B, B' are categories of our CG such that $A \Rightarrow B\ B'$, then B is a primitive category and $B' = B''/_{\mathrm{L}}B$ for some B''; in particular B' is non-primitive. (This is immediate from the structure of the possible categories we have assigned.) From this it readily follows by induction that, for any $A_1, \ldots, A_n$, there is at most one category A for which $A \overset{*}{\Rightarrow} A_1 \cdots A_n$, and only one parse tree corresponding to such a derivation. Indeed, to see this, let j be maximal such that A_j is primitive; then A_j must be a sister of A_{j+1} in the parse tree, and writing $B \Rightarrow A_j A_{j+1}$, we can apply an induction to $A \overset{*}{\Rightarrow} A_1 \cdots A_{j-1} B A_{j+2} \cdots A_n$.

Now we prove our claim, again by induction on n. The base case $n = 1$ is trivial since $A_1 = A$. Otherwise, A_1 must be primitive by the observation in the previous paragraph. Also, since we only use the left slash $/_{\mathrm{L}}$, we can see that $\tau(A_n) = A$; examining the categories our $\hat{f}$ assigns, we see that A_n must be of the form $A/_{\mathrm{L}}B$ or $(A/_{\mathrm{L}}B)/_{\mathrm{L}}B'$ for some B, B', hence is a right category. Now we know there exist some $j < k$ such that A_j is primitive and A_k is a right category, and we choose such a pair (j, k) with $k - j$ minimal. Then, for $j < l < k$, A_l is left and non-primitive, i.e. is of the form $B/_{\mathrm{L}}B'$ where B, B' are both primitive. It follows that in our parse tree, A_j, A_{j+1} must be sisters; their parent is a primitive category, so must be a sister of A_{j+2}; their parent is again primitive, and so forth. Thus, we see that there is a left-branching subtree that dominates the string $A_j A_{j+1} \cdots A_k$.

But from the type assignment rules, we can see that this forces

$$A_j = p_j \qquad A_{j+1} = (p_j, p_j, p_{j+1})/_{\mathrm{L}}p_j \qquad A_{j+2} = (p_j, p_{j+1}, p_{j+2})/_{\mathrm{L}}(p_j, p_j, p_{j+1}) \qquad \cdots$$

$$\cdots \qquad A_k = B/_{\mathrm{L}}(p_j, p_{k-2}, p_{k-1})$$

for some integers p_j and some B that is a left category of X_{p_j}. (Or, if $k = j + 1$, we have $A_j = p_j, A_{j+1} = B/_{\mathrm{L}}p_j$ and the argument will be similar.) We need to show that $X_{p_j} \overset{*}{\Rightarrow} U_j \cdots U_k$ under the context-free rules; the induction hypothesis applied to $A \overset{*}{\Rightarrow} A_1 \cdots A_{j-1} B A_{k+1} \cdots A_n$ will also give us that $A \overset{*}{\Rightarrow} U_1 \cdots U_{j-1} X_{p_j} U_{k+1} \cdots U_n$ in the context-free grammar, and combining will then give us the induction step, completing the proof.

But from the values of $A_j, \ldots, A_k$, it follows that $U_j = Y_{p_j}$; $U_l = Y_{p_l}$ and $X_{p_l} = Z_{p_{l-1}}$ for $j < l < k$; and $U_k = Z_{p_{k-1}}$. Hence, we have the context-free derivation

$$\begin{aligned} X_{p_j} \Rightarrow Y_{p_j} Z_{p_j} &= U_j X_{p_{j+1}} \Rightarrow U_j Y_{p_{j+1}} Z_{p_{j+1}} \\ &= U_j U_{j+1} X_{p_{j+2}} \Rightarrow \cdots \Rightarrow U_j \cdots U_{k-1} Z_{p_{k-1}} = U_j \cdots U_k, \end{aligned}$$

as required.

What we have shown is that, for terminals $A, U_1, \ldots, U_n \in V$, we have $A \overset{*}{\Rightarrow} U_1 \cdots U_n$ under the context-free rules if and only if there are categories A_i assigned to the U_i by $\hat{f}$ such that $A \overset{*}{\Rightarrow} A_1 \cdots A_n$ under the CG rules. But taking $A = S$, and considering strings of nonterminals that correspond to terminals (i.e. for which we have rules of the form $U_i \to a_i$), we then see that the CG generates precisely our context-free language, as needed.

Now we have shown that L is representable by a classical CG if L can be generated by a context-free grammar in Chomsky normal form. So we will be done if we can show that any context-free language not containing ϵ is generated by a context-free grammar in Chomsky normal form. This is standard ([33, ch. 4]); we normalize the grammar in a series of steps, each of which does not change the language it generates.

First, we may assume that every rule is of the form $X \to w$ where w is either a single terminal or a string of nonterminals. Indeed, for each terminal a, we can introduce a new nonterminal Y_a, and then simply replace every occurrence of a on the right side of a rule by Y_a, and also add the rule $Y_a \to a$.

Next, we may assume the right side of each rule contains at most two nonterminals. To accomplish this, we replace each rule of the form $X \to Y_1 Y_2 \cdots Y_n$ (where the Y_i are nonterminals) by rules $X \to Y_1 Z_1, Z_1 \to Y_2 Z_2, \ldots Z_{n-2} \to Y_{n-1} Z_{n-1}, Z_{n-1} \to Y_n$, where the Z_i are new nonterminal symbols.

Next, we will get rid of all the rules of the form $X \to \epsilon$. To do this, whenever we have a rule of the form $X \to YZ$ such that $Y \overset{*}{\Rightarrow} \epsilon$, we include a new rule $X \to Z$; if $Z \overset{*}{\Rightarrow} \epsilon$, we include a new rule $X \to Y$. This certainly does not enable us to derive any strings that were not already in the language. Now we can delete all the rules whose right-hand side is ϵ. To see that any string that was previously derivable is still derivable, just consider a parse tree for some string w under the old grammar; removing every node that dominates an empty string gives a parse tree for w under the new grammar (and this tree is nonempty, since $\epsilon \notin L$ means that $S \overset{*}{\not\Rightarrow} \epsilon$).

Now all our rules are of the forms $X \to a$, $X \to YZ$, or $X \to Y$ for nonterminals X, Y, Z, terminal a, and it remains to eliminate all rules of the last form. Well, if $X \overset{*}{\Rightarrow} a$, then we should include a rule $X \to a$, and if $X \overset{*}{\Rightarrow} YZ$, then we include a rule $X \to YZ$. Now we delete all rules of the form $X \to Y$. As in the previous step, this operation does not result in the loss of any possible strings: consider a parse tree for any string w in the old grammar; by contracting each maximal chain of nonbranching nodes to a single branching node, we get a parse tree for w in the new grammar. This gives our grammar in Chomsky normal form, and we are done. □

The remaining results we will prove in this appendix are from Section 6. First, we prove the basic theorem about unification of categories: any collection of sets that has a unifier has a most general unifier.

Proof of Theorem 15: Let C be a set of primitive categories and V the category space over C. We want to show that if a finite collection $\mathcal{S}$ of finite subsets of V has a unifier, then it has a most general unifier. Recall the definition of the length of a category, given inductively by $l(X) = 1$ for $X \in C$ and $l(X|Y) = l(X) + l(Y)$. We will here use the *pseudolength* given by $\hat{l}(X) = l(X) - 1$. We will prove the statement by induction first on the size of C (actually, the number of distinct elements of C that appear as a subtype of some element of some $D \in \mathcal{S}$) and then on the quantity

$$\lambda = \sum_{D \in \mathcal{S}} \left(\sum_{X \in D} \hat{l}(X) \right).$$

That is, in the induction step, we will reduce the statement to be proven either to a statement with a smaller C, or to one with the same C and a smaller value of λ.

First we need a base case: this will be when $\lambda = 0$. This happens if and only if every type in every $D \in \mathcal{S}$ is primitive. In this case, define a relation $\sim$ on the primitive types by $X \sim Y$ if and only if there is some $D \in \mathcal{S}$ with $X \in D, Y \in D$. $\sim$ induces an equivalence relation $\overset{*}{\sim}$ on C, by transitive closure. Let C' be the set of equivalence classes, and $\phi : C \to C'$ the map sending each X to its equivalence class. It is clear that the induced map ϕ of category spaces is a unifier of $\mathcal{S}$, and moreover any such unifier must factor through ϕ, so ϕ is in fact the most general unifier of $\mathcal{S}$.

Now, for the induction step, suppose $\lambda \neq 0$; then some $D \in \mathcal{S}$ contains a category of the form $X|Y$. We have three cases:

- If D contains another category of the form $X'|'Y'$ for $|' \neq |$, then $\mathcal{S}$ has no unifier (since there is no homomorphism that can send $X|Y, X'|'Y'$ to the same value), so we can go home.

- If every element of D is of the form $X'|Y'$ for some X', Y', then define the two sets $D_1 = \{X' \mid X'|Y' \in S\}$, $D_2 = \{Y' \mid X'|Y' \in S\}$. Notice that a homomorphism ϕ unifies D if and only if it unifies both D_1 and D_2, so if we let $\mathcal{S}'$ be the class obtained from $\mathcal{S}$ by replacing D with D_1 and D_2, then a most general unifier of $\mathcal{S}$ is the same thing as a most general unifier of $\mathcal{S}'$. Now $\hat{l}(X') + \hat{l}(Y') = \hat{l}(X'|Y') - 1$, and it follows that $\mathcal{S}'$ has a lower value for the parameter λ than $\mathcal{S}$ does (and uses the same C); hence we have performed the necessary reduction and can use the induction hypothesis.

- The only remaining case is if D contains a primitive category Z. If Z is a subtype of X or Y, then again D has no unifier and we are done, so suppose otherwise. In this case, let $C' = C \setminus \{Z\}$, with category space V', and define $\psi : V \to V'$ by $\psi(Z) = X|Y$ and $\psi(W) = W$ for all other $W \in C$. Let $\mathcal{S}' = \psi(\mathcal{S})$ (that is, each $D' \in \mathcal{S}$ is replaced by $\{\psi(W) \mid W \in D'\}$). We claim that ϕ is a unifier of $\mathcal{S}$ if and only if $\phi = \theta \circ \psi$ for some unifier θ of $\mathcal{S}'$. One direction is clear — that if θ unifies $\mathcal{S}'$, then $\theta \circ \psi$ unifies $\mathcal{S}$. Indeed, we only need to check that $\theta(\psi(X|Y)) = \theta(\psi(Z))$, but this is achieved since $\psi(X|Y) = X|Y = \psi(Z)$.

  So we need to prove the converse: if ϕ is a unifier of $\mathcal{S}$, then $\phi = \theta \circ \psi$ for some unifier θ of $\mathcal{S}'$. Well, put $\theta(W) = \phi(W)$ for each $W \in C'$. Then ϕ and $\theta \circ \psi$ agree for each W, and they also agree on Z since $\theta(\psi(Z)) = \theta(X|Y) = \phi(X|Y)$ (because $X|Y$ does not have Z as a subtype). Hence $\phi = \theta \circ \psi$. Now to check that θ is a unifier of $\mathcal{S}'$

just means (by construction of $\mathcal{S}'$) that $\theta(\psi(W)) = \theta(\psi(U))$ whenever W, U are in the same set of $\mathcal{S}$. But this is clear, because $\theta \circ \psi = \phi$, which is a unifier for $\mathcal{S}$.

Since $\mathcal{S}'$ uses the primitive category set C', a proper subset of C, we can again now apply the induction hypothesis, and the proof is done.

□

Next in line: the existence of general forms.

Proof of Lemma 18: First, construct several set-theoretically disjoint copies of the variable set Ω, one copy Ω_v for each internal node v of the rule structure T. Let C be the disjoint union of the Ω_v together with a copy of Σ. Let $\phi_v : \Omega \to C$ be the appropriate inclusion, which extends to a homomorphism of category spaces in the obvious way.

Each node in the tree will be relabeled with one or two categories over C:

- Whenever a node v is originally labeled (in T) with a rule schema $R : A \to A_1 \cdots A_n$, we relabel it with $\phi_v(A)$ and its daughters with $\phi_v(A_1), \ldots, \phi_v(A_n)$, respectively.

- A node that immediately dominates the terminal a will be relabeled a.

For each node v, let S_v be the set of its new labels, and let $\mathcal{S}$ be the family of all S_v as v varies. Also define a grammar G_0 whose set of primitive categories C_0 consists of all elements of C occurring as a subpart of any new node label; whose start symbol is $\phi_r(A)$, where r is the root node and A is the left side of the rule schema applying there (which necessarily is a single variable); and whose category assignment function simply assigns each $a \in \Sigma$ occurring on a leaf of T to the category a itself, and assigns no categories to any other terminal.

Now it is straightforward to check the following: For any rigid grammar G for which T is a valid rule structure, G is obtained from some $\phi(G_0)$, where ϕ is a unifier of $\mathcal{S}$, by possibly adding some more terminal category assignments. And conversely, if ϕ is a unifier of $\mathcal{S}$, then T is a valid rule structure for the grammar $\phi(G_0)$; the homomorphism ϕ simply tells us exactly how to instantiate the rules at each node. Hence, if we let ϕ be the most general unifier of $\mathcal{S}$, then $\phi(G_0)$ is the desired general form for T. □

Finally, we get to something interesting: proving the finite elasticity of classical CGs. The proof is somewhat involved. Here it is.

Proof of Lemma 21: In order to prove this statement, we need to introduce a couple new ideas concerning category spaces. If V is a category space over the set of primitive categories C, and $K \subseteq V$, let the *span* $s(K)$ denote the set of all elements of V obtained from elements of K by repeatedly applying operations in O, and let the *rank* $r(K)$ be the size of the smallest $D \subseteq V$ such that $K \subseteq s(D)$. Thus, for example, every subset $K \subseteq V$ has rank at most $|C|$ (just take $D = C$), and also rank at most $|K|$ (just take $D = K$). V itself has rank exactly $|C|$ (since we must have $C \subseteq D$; there is no way to get the primitive categories by applying operations of O to other categories).

We have the following basic property:

Sublemma 25 *If V, V' are category spaces and $\phi : V \to V'$ a homomorphism, then we have $r(\phi(V)) \leq r(V)$. If ϕ is not one-one, then $r(\phi(V)) < r(V)$.*

Proof: Let C be the set of primitive categories for V. The first statement is obvious since $\{\phi(Q) \mid Q \in C\}$ spans $\phi(V)$. We prove the second statement by induction on $l(X) + l(Y)$, where X, Y are distinct categories such that $\phi(X) = \phi(Y)$. For the base case, both X and Y are primitive. Then $\phi(V)$ is generated by $\phi(X)$ together with the images of the other $|C| - 2$ primitive categories, so it has rank at most $|C| - 1$.

Otherwise, we may assume $X = U|W$ (for some $| \in O$). If Y is not primitive, then write $Y = U'|'W'$. We get $\phi(U)|\phi(W) = \phi(U')|'\phi(W')$, so $\phi(U) = \phi(U')$ and $\phi(W) = \phi(W')$. Since $X \neq Y$, either $U \neq U'$ or $W \neq W'$; either way, we can apply the induction hypothesis.

Finally, if $X = U|W$ but Y is primitive, then Y cannot be a subtype of X (otherwise $\phi(Y)$ would be a proper subtype of $\phi(X)$, and the two could not be equal). So if $\hat{C} = C \setminus \{Y\}$ and $\hat{V}$ is the category space over $\hat{C}$, then all primitive categories are mapped by ϕ into $\phi(\hat{V})$ (this is trivial for primitive categories other than Y, and for Y it holds because $\phi(Y) = \phi(X)$ and $X \in \hat{V}$). Thus

$$r(\phi(V)) = r(\phi(\hat{V})) \leq r(\hat{V}) = |\hat{C}| < |C| = r(V).$$

□

Now, for any grammar G, let $U(G)$ be the set of all categories that appear in any S-parse tree of G. Also say that G *uses* a terminal a if a appears in some string in $L(G)$. We use the following sublemma:

Sublemma 26 *Suppose G is a categorial grammar over Σ that uses every terminal, and ϕ a homomorphism such that $S(\phi(G))$ contains a structure not in $S(G)$ (where we are using the classical rule schemata (8)). Then $r(U(\phi(G))) < r(U(G))$.*

Proof: We may assume that C is the set of primitive categories of V and $r(U(G)) = |C|$, since otherwise we can let $D \subseteq V$ be a set of minimal size with $U(G) \subseteq s(D)$, and rewrite the elements of $U(G)$, treating the elements of D as primitive categories. We may also assume that ϕ is one-one, since otherwise Sublemma 25 applies and we are done.

Let $K = U(\phi(G))$. Notice that $\phi(U(G))$ is a proper subset of K: otherwise, for each S-parse tree of $\phi(G)$, we could apply ϕ^{-1} to each node label and obtain an S-parse tree of G, contradicting the assumption $S(G) \subset S(\phi(G))$ (this step depends crucially on the fact that ϕ is one-one, so that $\phi^{-1}(X|Y) = \phi^{-1}(X)|\phi^{-1}(Y)$). Also let $D_0 = \{\phi(Q) \mid Q \in C\}$, so $|D_0| \leq r(U(G))$. We will construct a finite sequence of sets

$$D_0, D_1, \ldots, D_m \subseteq V'$$

such that we have strict inclusions

$$(s(D_0) \cap K) \subset (s(D_1) \cap K) \subset \cdots \subset (s(D_m) \cap K) = K.$$

These sets will satisfy $|D_{i+1}| \leq |D_i|$, and $|D_{i+1}| < |D_i|$ for at least one i. Then we will be done, since we will have

$$r(K) \leq |D_m| < |D_0| \leq |C| = r(U(G)).$$

The construction is inductive, so suppose D_i is given, with $K \not\subseteq s(D_i)$. We know, by the argument in the proof of Theorem 4, that K is a finite set. So let X be an element of

$K \setminus s(D_i)$ that occurs as low as possible in some S-parse tree T of $\phi(G)$; if there are multiple such choices for X, choose one that maximizes the length $l(X)$. (We will be somewhat sloppy here about the distinction between a node and its label, but this should cause no confusion.) We know that X cannot be a category assigned to a terminal in $\phi(G)$, since otherwise it would be in $s(D_0) \cap K \subseteq s(D_i)$; hence it must be a branching node, with daughters of the forms $X|Y$ and Y (here of course $| \in \{/_{\mathrm{L}}, /_{\mathrm{R}}\}$). By assumption $X|Y \in s(D_i)$ and $Y \in s(D_i)$. $X|Y$ must be an element of D_i itself, since otherwise we would have $X \in s(D_i)$. So let $D_i' = D_i \cup \{X\} \setminus \{X|Y\}$. Then X and Y are in $s(D_i')$, so $X|Y \in s(D_i')$ and hence $s(D_i) \subset s(D_i')$. Their intersections with K maintain this strict subsethood, since X is in $s(D_i')$ but not $s(D_i)$.

Now, if $s(D_i')$ does not contain all the node labels of T, then put $D_{i+1} = D_i'$. If on the other hand $s(D_i')$ does contain all the node labels of T, we will go a bit further. Since X cannot be the root node label (otherwise $X = \phi(S) \in D_0 \cap K \subseteq s(D_i)$), it must have a mother and a sister. Let the mother be labeled U; it then has two daughters, of the forms W, $U|'W$, one of which is equal to X. We also, by assumption, have $U, W \in s(D_i')$. If $X = U|'W$, then $X \in D_i'$ by construction. If $X = W$, then our choice of X (as maximizing $l(X)$ for given depth) implies $U|'W \in s(D_i)$, hence in fact $U|'W \in D_i$ because otherwise we would have $X \in s(D_i)$. Moreover, $U|'W \neq X|Y$, because otherwise we would have $X = W = Y$, contradicting $Y \in s(D_i)$. Hence, $U|'W$ is still in D_i'. So regardless of which of the two daughters of U is equal to X, we have $U|'W \in D_i'$. But since $U, W \in s(D_i')$, we can let $D_{i+1} = D_i' \setminus \{U|'W\}$, and we still have $s(D_{i+1}) = s(D_i')$; meanwhile $|D_{i+1}| < |D_i'| = |D_i|$, as needed.

Figure 4 shows the case $X = W$ of this argument. The labels that are necessarily in $s(D_i)$ are circled; those that are necessarily in D_i itself are circled thickly.

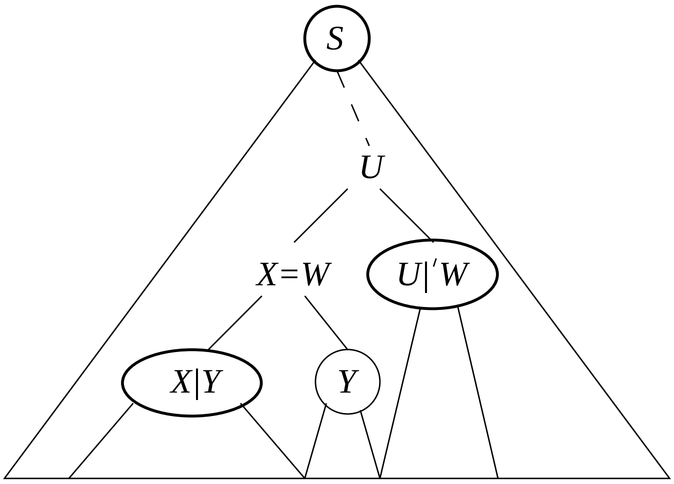


Figure 4: The proof of Sublemma 26

Thus we have the inductive construction of D_{i+1} from D_i. We have shown that $s(D_i) \cap K \subseteq s(D_{i+1}) \cap K$, and in fact the inclusion is strict since $X \in s(D_{i+1}) \setminus s(D_i)$. Because K is finite, this process can be repeated only finitely many times, and eventually we get some D_m with $K \subseteq s(D_m)$. We have shown that at each stage $|D_{i+1}| \leq |D_i|$. Moreover, if we consider the latest i such that $U(G) \not\subseteq s(D_i)$, then in the course of constructing D_{i+1} from D_i, we exhaust the labels on some parse tree; hence for this i, $|D_{i+1}| < |D_i|$. This finishes the proof.

□

Now back to our main goal. Suppose that $G_1 \sqsubseteq G_2 \sqsubseteq \cdots$ are rigid categorial grammars with the classical rule schema, and

$$S(G_1) \subset S(G_2) \subset \cdots$$

are all strict inclusions. If $G \sqsubseteq G'$ then every terminal used by G is used by G'; since there are only finitely many terminals, it follows that G_n must use the same set of terminals for all sufficiently large n. So, ignoring finitely many initial terms, we may assume that all G_n use the same set of terminals. It follows from the definition of $\sqsubseteq$ that there exist homomorphisms $\phi_1, \phi_2, \ldots$ of the category space such that $G_{i+1} = \phi_i(G_i)$ for each i. But since the inclusions of structure languages are strict, Sublemma 26 tells us that $r(U(G_{i+1})) < r(U(G_i))$. Hence the ranks of the $U(G_i)$ form a strictly decreasing sequence of positive integers, which is a contradiction. □

It is worth taking at least a few moments to discuss other rule schemata that can be added to Φ while preserving finite elasticity. The crucial step is Sublemma 26, since it is the only part of the proof that makes any reference to the rule schemata. The requirements we must impose on the rule schemata in order for the preceding proof to go through are fairly strict. They must be depth-preserving, since we have made use of the finiteness of the set K. They must also have the property that, if ϕ is a one-one homomorphism of category spaces, then applying ϕ^{-1} to each term of a ground instance (if possible) gives a ground instance. Thus, for example, the composition rule $X/_{\mathrm{R}}Z \to X/_{\mathrm{R}}Y\ Y/_{\mathrm{R}}Z$ fails to meet this criterion, because a category space with categories I, J, K and the one-one homomorphism given by

$$\phi(\mathrm{I}) = \mathrm{I}/_{\mathrm{R}}\mathrm{J}, \qquad \phi(\mathrm{J}) = \mathrm{J}/_{\mathrm{R}}\mathrm{K}, \qquad \phi(\mathrm{K}) = \mathrm{I}/_{\mathrm{R}}\mathrm{K}$$

has the property that $\phi(\mathrm{K}) \to \phi(\mathrm{I})\ \phi(\mathrm{J})$ is a ground instance, but $\mathrm{K} \to \mathrm{I}\ \mathrm{J}$ is not.

But the further properties needed in order to make the proof succeed are subtler. We can at least add rules of the form

$$X \to \text{some ordering of } X_1, \ldots, X_n, B \tag{36}$$

where $X, X_1, \ldots, X_n$ are variables, and B is a category in $s(X, X_1, \ldots, X_n)$ having each variable $X, X_1, \ldots, X_n$ as a subtype. The proof goes through essentially as before, with two changes:

- in forming D_i' from D_i, instead of removing $X|Y \in D_i$, we only know that some subtype of this instance of B is an element of D_i having X as a subtype, and this is the one we remove;
- in the case where all the labels of T are in $s(D_i')$, we do not necessarily know exactly how to remove one element to form our D_{i+1}, but we can still show that some element (our node X or one of its sisters) can be obtained from D_i' using operations of O in two different ways, and then we can apply Sublemma 25 to show that there exists some D_{i+1} with $|D_{i+1}| < |D_i'|$, $s(D_i') \subseteq s(D_{i+1})$.

Kanazawa ([43, ch. 9]) notes that adding rules such as

$$X \to X\ X \tag{37}$$

also does not get in the way of finite elasticity. The same proof goes through, except that instead of choosing X as low as possible, we must choose an X node that does not arise on the right-hand side of an instance of (37) and is as low as possible given this constraint.

Anyhow, finding more rules that preserve the finite elasticity property is an interesting mathematical recreation, but it is not our prime concern here.

References

[1] A. Ades and M. Steedman, "On the Order of Words," *Linguistics and Philosophy* **4**, 1982: 517-558.

[2] A. V. Aho, "Indexed grammars: An extension to context-free grammars," *Journal of the ACM* **15**, 1968: 647-671.

[3] K. Ajdukiewicz, "Die syntaktische Konnexität," *Studia Philosophica* **1**, 1935: 1-27.

[4] D. Angluin, "Inductive Inference of Formal Languages from Positive Data," *Information and Control* **45**, 1980: 117-135.

[5] D. Angluin, "Finding Patterns Common to a Set of Strings," *Journal of Computer and System Sciences* **21**, 1980: 46-62.

[6] C. L. Baker and J. McCarthy, eds., *The Logical Problem of Language Acquisition* (Cambridge: MIT Press), 1981.

[7] J. Baldridge, *Lexically Specified Derivational Control in Combinatory Categorial Grammar*, Ph. D. thesis (University of Edinburgh, School of Informatics), 2002.

[8] Y. Bar-Hillel, "A Quasi-Arithmetical Notation for Syntactic Description," *Language* **29**, 1953: 47-58, reprinted in Y. Bar-Hillel, *Language and Information* (Reading: Addison-Wesley), 1964.

[9] Y. Bar-Hillel, H. Gaifman, and E. Shamir, "On categorial and phrase structure grammars," *The Bulletin of the Research Council of Israel* **9F**, 1960: 1-16.

[10] R. Berwick and S. Epstein, "On the Convergence of 'Minimalist' Syntax and Categorial Grammar," in A. Nijholt, G. Scollo, and R. Steetkamp, eds., *Algebraic Methods in Language Processing 1995: Proceedings of the Twente Workshop on Language Technology 10* (Enschede: Universiteit Twente), 1995.

[11] J. Bresnan, R. Kaplan, S. Peters, and A. Zaenen, "Cross-Serial Dependencies in Dutch," *Linguistic Inquiry* **13** (4), 1982: 613-635, reprinted in W. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, eds., *The Formal Complexity of Natural Language* (Dordrecht: Reidel), 1987.

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

[13] W. Buszkowski, "Generative Power of Categorial Grammars," in R. Oehrle, E. Bach, and D. Wheeler, eds., *Categorial Grammars and Natural Language Structures* (Dordrecht: Reidel), 1988: 69-94.

[14] W. Buszkowski and G. Penn, "Categorial Grammars Determined from Linguistic Data by Unification," *Studia Logica* **49**, 1990: 431-454.

[15] S. Carey, *The Origin of Concepts* (Cambridge: MIT Press), forthcoming.

[16] N. Chomsky, *Aspects of the Theory of Syntax* (Cambridge: MIT Press), 1965.

[17] N. Chomsky, "Introduction to the Formal Analysis of Natural Languages," in R. Luce, R. Bush, E. Galanter, eds., *Handbook of Mathematical Psychology, Volume II* (New York: Wiley and Sons), 1967: 269-322.

[18] N. Chomsky, *Syntactic Structures* (The Hague: Mouton), 1969.

[19] N. Chomsky, *Logical Structure of Linguistic Theory* (New York: Plenum Press), 1975.

[20] N. Chomsky, *Lectures on Government and Binding* (Dordrecht: Foris), 1981.

[21] N. Chomsky, *Knowledge of language: its nature, origin, and use* (Westport: Praeger), 1986.

[22] N. Chomsky, *The Minimalist Program* (Cambridge: MIT Press), 1995.

[23] C. Culy, "The Complexity of the Vocabulary of Bambara," *Linguistics and Philosophy* **8**, 1985: 345-351, reprinted in W. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, eds., *The Formal Complexity of Natural Language* (Dordrecht: Reidel), 1987.

[24] H. B. Curry and R. Feys, *Combinatory Logic, Volume I* (Amsterdam: North Holland), 1958.

[25] D. Dowty, "Grammatical Relations and Montague Grammar," in P. Jacobson and G. Pullum, eds., *The Nature of Syntactic Representation* (Dordrecht: Reidel Publishing), 1982.

[26] J. Feldman, "Some Decidability Results on Grammatical Inference and Complexity," *Information and Control* **20**, 1972: 244-262.

[27] M. Fitting, *First-Order Logic and Automated Theorem Proving* (Berlin: Springer), 1996.

[28] J. Fodor, *Modularity of mind: an essay on faculty psychology* (Cambridge: MIT Press), 1983.

[29] G. Gazdar, "Applicability of Indexed Grammars to Natural Languages," in U. Reyle and C. Rohrer, eds., *Natural Language Parsing and Linguistic Theories* (Dordrecht: Reidel), 1988: 69-94.

[30] G. Gazdar, G. Pullum, R. Carpenter, E. Klein, T. Hukari, and T. Levine, "Category structures," *Computational Linguistics* **14** (1), 1988: 1-19.

[31] E. M. Gold, "Language identification in the limit," *Information and Control* **10**, 1967: 447-474.

[32] G. Grätzer, *Lattice Theory: First concepts and distributive lattices* (San Francisco: W. H. Freeman), 1971.

[33] M. Harrison, *Introduction to Formal Language Theory* (Reading: Addison-Wesley), 1978.

[34] M. Hauser, *Wild Minds: What Animals Really Think* (New York: Holt and Company), 2000.

[35] B. Hoffman, "The Formal Consequences of Using Variables in CCG Categories," in *Proceedings of the 31st Meeting of the Association for Computational Linguistics*, 1993: 298-300.

[36] N. Hornstein and D. Lightfoot, eds., *Explanation in Linguistics: The logical problem of language acquisition* (London: Longman), 1981.

[37] M. A. C. Huybregts, "Overlapping Dependencies in Dutch," *Utrecht Working Papers in Linguistics* I, 1976: 24-65.

[38] P. Jacobson, *Introduction to Syntax and Semantics: A Categorial Grammar Approach*, in preparation.

[39] A. K. Joshi, L. S. Levy, and M. Takahashi, "Tree adjunct grammars," *Journal of Computer and System Sciences* **19** (1), 1975: 136-163.

[40] A. K. Joshi, "Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?" in D. Dowty, L. Karttunen, and A. Zwicky, eds., *Natural Language Parsing: Psychological, computational, and theoretical perspectives* (Cambridge: Cambridge Press), 1985: 206-250.

[41] A. K. Joshi, K. Vijay-Shanker, and D. Weir, "The convergence of mildly context-sensitive grammar formalisms," in P. Sells, S. Shieber, and T. Wasow, eds., *Foundational Issues in Natural Language Processing* (Cambridge: MIT Press), 1991.

[42] M. B. Kac, "Surface Transitivity, *Respectively* Coordination, and Context-Freeness," *Natural Language and Linguistic Theory* **5**, 1987: 441-452.

[43] M. Kanazawa, *Learnable Classes of Categorial Grammars* (Stanford: CSLI Publications), 1998.

[44] J. Michaelis, "Derivational minimalism is mildly context-sensitive," in M. Moortgat, ed., *Logical Aspects of Computational Linguistics*, Lecture Notes in Computer Science vol. 2014 (Berlin: Springer), 2001.

[45] G. Morrill, *Type Logical Grammar: Categorial Logic of Signs* (Dordrecht: Kluwer Academic Publishers), 1994.

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

[47] D. Muller and P. Schupp, "The theory of ends, pushdown automata, and second-order logic," *Theoretical Computer Science* **37**, 1985: 51-75.

[48] M. A. Palis and S. M. Shende, "Pumping Lemmas for the Control Language Hierarchy," *Mathematical Systems Theory* **28**, 1995: 199-213.

[49] B. Partee, A. ter Meulen, and R. Wall, *Mathematical Methods in Linguistics* (Dordrecht: Kluwer Academic Publishers), 1993.

[50] T. Peters and R. W. Ritchie, "On restricting the base component of transformational grammars," *Information and Control* **18**, 1971: 483-501.

[51] T. Peters and R. W. Ritchie, "On the generative power of transformational grammars," *Information Sciences* **6**, 1973: 49-83.

[52] S. Pinker and P. Bloom, "Natural language and natural selection," *Behavioral and Brain Sciences* **13**, 1990: 707-727.

[53] S. Pinker, *The Language Instinct* (New York: HarperCollins), 1994.

[54] P. M. Postal, *On Raising: One Rule of English Grammar and its Theoretical Implications* (Cambridge: MIT Press), 1974.

[55] G. Pullum and G. Gazdar, "Natural Languages and Context-Free Languages," *Linguistics and Philosophy* **4** (1982): 471-504, reprinted in W. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, eds., *The Formal Complexity of Natural Language* (Dordrecht: Reidel), 1987.

[56] D. Radzinski, "Chinese Number-Names, Tree Adjoining Languages, and Mild Context Sensitivity," *Computational Linguistics* **17** (3), 1991: 277-299.

[57] K. Roach, "Formal properties of head grammars," in A. Manaster-Ramer, *Mathematics of Language* (Amsterdam: Benjamins), 1987: 293-348.

[58] W. C. Rounds, A. Manaster-Ramer, and J. Friedman, "Finding Natural Languages a Home in Formal Language Theory," in A. Manaster-Ramer, *Mathematics of Language* (Amsterdam: Benjamins), 1987: 349-359.

[59] A. Salomaa, *Formal Languages* (New York: Academic Press), 1973.

[60] W. Savitch, “Theories of Language Learnability,” in A. Manaster-Ramer, ed., *Mathematics of Language* (Amsterdam: Benjamins), 1987.

[61] S. Shieber, *An Introduction to Unification-Based Approaches to Grammar* (Stanford: CSLI), 1986.

[62] S. Shieber, “Evidence Against the Context-Freeness of Natural Language,” *Linguistics and Philosophy* **8** (1985): 333-343, reprinted in W. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, eds., *The Formal Complexity of Natural Language* (Dordrecht: Reidel), 1987.

[63] E. Stabler, “Derivational minimalism,” in C. Retoré, ed., *Logical Aspects of Computational Linguistics*, Lecture Notes in Computer Science vol. 1328 (Berlin: Springer), 1997.

[64] E. Stabler, “The 3-D lexical matrix,” presentation at Program in Evolutionary Dynamics, Harvard University, March 2004.

[65] M. Steedman, “Dependency and Coordination in the Grammar of Dutch and English,” *Language* **61** (3), 1985: 523-568.

[66] M. Steedman, “Combinatory Grammars and Parasitic Gaps,” *Natural Language and Linguistic Theory* **5**, 1987: 403-439.

[67] M. Steedman, “Combinators and Grammars,” in R. Oehrle, E. Bach, and D. Wheeler, eds., *Categorial Grammars and Natural Language Structures* (Dordrecht: Reidel), 1988: 417-442.

[68] M. Steedman, *The Syntactic Process* (Cambridge: MIT Press), 2000.

[69] M. Steedman and J. Baldridge, “Combinatory Categorial Grammar,” in preparation, at `ftp://ftp.cogsci.ed.ac.uk/pub/steedman/ccg/manifesto.pdf` (accessed 8 January 2005).

[70] L. Valiant, “A Theory of the Learnable,” *Communications of the ACM* **27** (11), 1984: 1134-1142.

[71] K. Vijay-Shanker, *A study of tree adjoining grammars*, Ph.D. thesis (University of Pennsylvania), 1987.

[72] K. Vijay-Shanker and D. Weir, “Characterizing structural descriptions produced by various grammatical formalisms,” *25th Meeting of the Association for Computational Linguistics*, 1987: 104-111.

[73] K. Vijay-Shanker and D. Weir, “The equivalence of four extensions of context-free grammar,” *Mathematical Systems Theory* **27**, 1994: 511-546.

[74] D. Weir, *Characterizing Mildly Context-Sensitive Grammar Formalisms*, Ph.D. thesis (University of Pennsylvania), 1988.

[75] D. Weir and A. K. Joshi, "Combinatory categorial grammars: Generative power and relationship to linear context-free rewriting systems," in *Proceedings of the 26th Meeting of the Association for Computational Linguistics*, 1988: 278-285.

[76] D. Weir, "A geometric hierarchy beyond context-free languages," *Theoretical Computer Science* **104**, 1992: 235-261.

[77] M. M. Wood, *Categorial Grammars* (London: Routledge), 1993.

[78] K. Wright, "Identification of unions of languages drawn from an identifiable class," in *The 1989 Workshop on Computational Learning Theory* (San Mateo: Morgan Kaufmann), 1989: 328-333.

[79] T. Yokomori, "Polynomial-time identification of very simple grammars from positive data," *Theoretical Computer Science* **298**, 2003: 179-206.

Index