CONTEXT-FREE LANGUAGES. The context-free languages are perhaps the most important class of formal languages (i.e., sets of symbol strings) for both linguistics and computer science. There are numerous equivalent characterizations of the class.

The standard formalization is in terms of a type of rewriting system known as a *context-free phrase structure grammar* (henceforth CFG). CFGs were introduced by Noam Chomsky in the 1950s to reconstruct the practice of much earlier traditional and structuralist syntactic description (see Chomsky 1959 for a mathematical treatment) — though Chomsky did not use the term "context-free," calling the relevant systems *Type 2 grammars*.

A CFG is a system of rules that, intuitively, say things like 'a noun phrase may consist of an article followed by a noun', or 'a complement clause may consist of the word *that* followed by a finite clause', or 'an intransitive verb may consist solely of an instance of the word *expire*'. Such rules may be given in the form $\alpha \rightarrow \varphi$, where α being a single nonterminal (i.e., a category symbol like NP for 'noun phrase') and φ is a string of zero or more symbols that may be either nonterminals or terminals (i.e., words in the language). The standard interpretation of rules of this kind (another interpretation is discussed below) is as an instruction to take a string $X\alpha Y$ (where X and Y can be anything, and are not specified as part of the rule) and change it to $X\varphi Y$.

When such rules are applied to a special nonterminal known as the *start symbol*, and reapplied to the resultant string recursively, in some cases a string entirely composed of terminals will be derived. A CFG *generates*, or defines as well formed, all and only those strings of terminals that can be derived through some combination of applications of the rewriting rules. A set of strings of terminals is a *context-free language* (henceforth, CFL) iff there is some CFG that generates exactly that set.

To give a simple example, CFGs can define mirror-image structure: the set of all even-length letter strings over some finite alphabet in which the second half is an exact repetition of the first in the opposite order (i.e., the set of all palindromes over the relevant alphabet) is a CFL. To take another example, if we take the left and right parentheses '`(`' and '`)`' as our terminals and we assume the five rules $\{S \rightarrow LR, S \rightarrow LSR, S \rightarrow SS, L \rightarrow (, R \rightarrow)\}$, with S as the start symbol, then the set of strings generated is the set of all properly matched strings of parentheses, such as '`((()))()(())`', so that language is also a CFL.

To take a more interesting example, the set of all properly formed HTML

documents (web page files) with arbitrary text is a CFL. An HTML document (with arbitrary text content) has this sort of structure:

```
<HTML> <HEAD> <TITLE>Jane Doe's Home Page</TITLE> </HEAD>
<BODY> <H1>Jane Doe</H1> <H2>Home Page</H2> <P>
<CENTER><IMG src="jane.jpg"></CENTER></P> </BODY> </HTML>
```

The expression `<HTML>` must be followed by `</HTML>`, `<HEAD>` must be followed by `</HEAD>`, and so on, in the same pattern as matched parentheses. Thus recognizing that a string belongs to a certain CFL is one of the tasks performed by a web browser.

The CFLs may be characterized in a way that is independent of grammars. → *Automata* are abstract machines defined to accept certain inputs and to fail to accept others. A *nondeterministic pushdown stack automaton* (NPDA) is a nondeterministic automaton with a 'last-in, first-out' memory access. A set of strings if a CFL iff there is some NPDA that accepts all and only the strings in that set. (This result was obtained in the early 1960s by N. Chomsky and M. P. Schützenberger, and independently by R. J. Evey). Deterministic pushdown automata (DPDAs), which in every configuration have only one move that is legal, define a different and smaller class of languages, the *deterministic CFLs*, which have considerable importance in computer science.

CFG rules do not have to be interpreted as defining rewriting operations on strings. McCawley (1968) suggested that a rule '$A \rightarrow BC$' can be interpreted as a template that is matched by pieces of *trees*, namely any pieces consisting of a node labeled A that has two daughters, the left labeled B and the right labeled C. We can call a tree admissible iff every node in it is either labeled with a terminal or matches a rule. Under this interpretation, a rule is really just a local tree (a mother node together with its daughters), and a tree is admissible iff it is entirely composed of local trees that are members of the set. If every node with no daughters is labelled with a terminal symbol, the sequence of such nodes from left to right is called the *yield* of a tree. Given a set of trees, their yields define a language (which may be empty). A language is a CFL iff it is the yield of a set of trees containing all and only finite trees admissible by some CFG.

Surprisingly, the result continues to hold even if we allow context-sensitive rules such as $A \rightarrow BC/D\_E$ in defining the admissibility of local trees: a node in a tree labeled A with daughters labeled respectively B and C is admissible if there is a node labeled D to its left in the tree and a node labeled E to

its right. The yield of all finite trees that are admissible according to some *context-sensitive* grammar is still a CFL. (The original proof of this result was by P. Stanley Peters and Robert W. Ritchie (1969[1973]), following up on McCawley's work.)

A closely related characterization of the CFLs uses finite state tree automata, which read trees and check the node labels. A tree is accepted by a tree automaton iff every node in it is admissible according to the automaton's rules, and a set of trees is accepted if all and only the members of the set are accepted. A set of strings is a CFL iff it is the yield of a set of trees accepted by some *finite state* tree automaton.

Work in computer science around 1970 led to a result that connects the CFLs, via tree automata, to a particular kind of logic. A logical language can be given a semantics in which the models are trees; and the language can be appropriately designed for making statements about trees in the linguist's sense, with predicates for expressing notions like 'is labeled A', relation symbols for 'dominates' and 'precedes', and the usual quantifiers, variables, and connectives. Such a language is *first order* if its quantifiers are the usual ones $\forall$ and $\exists$, and its variables range only over nodes. A language that has variables ranging over both nodes and finite sets of nodes is a *weak monadic second order* language ('monadic' because one-place predicates can be quantified over but polyadic relations cannot; weak because only finite sets can be quantified over). A tree is said to *satisfy* a sentence of a logical language if and only if everything the sentence entails is true of that tree. It was shown around 1970 that a set of strings is a CFL iff it is the yield of a set T of finite trees, and there is a sentence ψ of a monadic second-order logical language with trees as its intended models such that ψ is satisfied by all and only and only the members of T. This gives a characterization of the CFLs in terms that make no reference to either grammars or automata (see Rogers 1998 for a useful introduction to this way of looking at syntactic description).

The theory of CFLs is extremely important for theoretical linguistics. A number of theories of syntax not originally designed to be equivalent to context-free phrase structure grammars later turned out to be equivalent to it. Postal (1964) claims to show that essentially all structuralist syntactic theories were equivalent to context-free phrase structure grammar in terms of the sets of strings they could define. It was shown in the 1960s that several types of categorial grammars generate only CFLs, and this result has since been extended to Lambek-style grammars (Pentus 1993). Virtually all parsing technology for natural languages has been built around different

techniques for parsing CFLs (sometimes deterministic CFLs, in the sense defined above in connection with pushdown stack automata). When a parser intended to work on transformational grammars of a certain sort was devised by Marcus (1980), it was later proved by Nozohoor-Farshi (1984) that such a parser could only parse CFLs.

Postal (1964) attempted to show not only that almost all previous syntactic theories were equivalent to CFG, but also that Mohawk was not a CFL, so that those earlier theories were false and the additional power of transformational grammars was necessary. However, the argument for the non-CFL character of Mohawk was flawed both formally and empirically; so were all arguments of the same sort up till the early 1980s (see Pullum and Gazdar 1982). Generalized phrase structure grammar as described in Gazdar, Klein, Pullum and Sag (1985) limited itself entirely to context-free description yet was able to describe a very broad range of English syntactic constructions. It was nonetheless eventually shown that the syntax of natural languages cannot always be appropriately modeled in terms of CFLs. Certain phenomena have been found in natural languages that are beyond the power of context-free → *generative capacity*.

In current work on → *natural language processing* and → *automatic speech recognition*, stochastic CFGs are very important. A stochastic grammar has a probability associated with each rule, and generates strings paired with probabilities (for an introduction see Jelinek 1998).

References Chomsky, Noam (1959). On certain formal properties of grammars. *Information and Control* **2** 137–167

Chomsky, N. and M.P. Schützenberger (1963). The algebraic theory of context-free languages. In: P. Braffort and D. Hirschberg (eds): *Computer Programming and Formal Languages* 118–161 Amsterdam: North Holland

Evey, R.J. (1963). The theory and application of pushdown store machines. In: *Mathematical Linguistics and Automatic Translation*, NSF-IO, 217–255. Cambridge, MA: Harvard University

Gazdar, Gerald; Ewan Klein; Geoffrey K. Pullum; and Ivan A. Sag (1985). *Generalized Phrase Structure Grammar*. Oxford: Basil Blackwell.

Jelinek, Frederic (1988) *Speech Recognition by Statistical Methods*. Cambridge MA: MIT Press

Marcus, Mitchell P. (1980). *A Theory of Syntactic Recognition for Natural Language*. Cambridge, MA: MIT Press.

McCawley, James D. (1968) Concerning the base component of a transformational-generative grammar. *Foundations of Language* 4, 243–269.

Nozohoor-Farshi, R. (1984) Context-freeness of the language accepted by Marcus' parser. *25th annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, 117–122. Menlo Park, CA: Association for Computational Linguistics.

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

Peters, P. S. and R. W. Ritchie. Context-sensitive immediate constituent analysis — context-free languages revisited. *Proceedings of the ACM Symposium on Theory of Computing*, 1–8. Also in *Mathematical Systems Theory* 6, 324–333.

Postal, Paul M. (1964) *Constituent Structure: A Study of Contemporary Models of Syntactic Description.* Bloomington, IN: Publication 30, Indiana University Research Center in Anthropology, Folklore and Linguistics.

Pullum, Geoffrey K. and Gerald Gazdar (1982) Natural languages and context-free languages. *Linguistics and Philosophy* 4, 471–504.

Rogers, James. 1998. *A Descriptive Approach to Language-Theoretic Complexity.* Stanford, CA: CSLI Publications.

Index context free grammar, context free language, deterministic context free language, pushdown automata, tree automata, node admissibility, monadic second order predicate, categorial grammar, Lambek grammar