Chomsky Hierarchy

Authored by

mohammad looti

November 19, 2022

Within the field of computer science, specifically in the area of formal languages, the Chomsky hierarchy (occasionally referred to as Chomsky-Schützenberger hierarchy) is a containment hierarchy of classes of formal grammars.

This hierarchy of grammars was described by Noam Chomsky in 1956. It is also named after Marcel-Paul Schützenberger who played a crucial role in the development of the theory of formal languages.

Formal grammars

A formal grammar of this type consists of:

a finite set of terminal symbols
a finite set of nonterminal symbols
a finite set of production rules with a left and a right-hand side consisting of a sequence of these symbols
a start symbol

A formal grammar defines (or generates) a formal language, which is a (usually infinite) set of finite-length sequences of symbols (i.e. strings) that may be constructed by applying production rules to another sequence of symbols which initially contains just the start symbol. A rule may be applied to a sequence of symbols by replacing an occurrence of the symbols on the left-hand side of the rule with those that appear on the right-hand side. A sequence of rule applications is called a derivation. Such a grammar defines the formal language: all words consisting solely of terminal symbols which can be reached by a derivation from the start symbol.

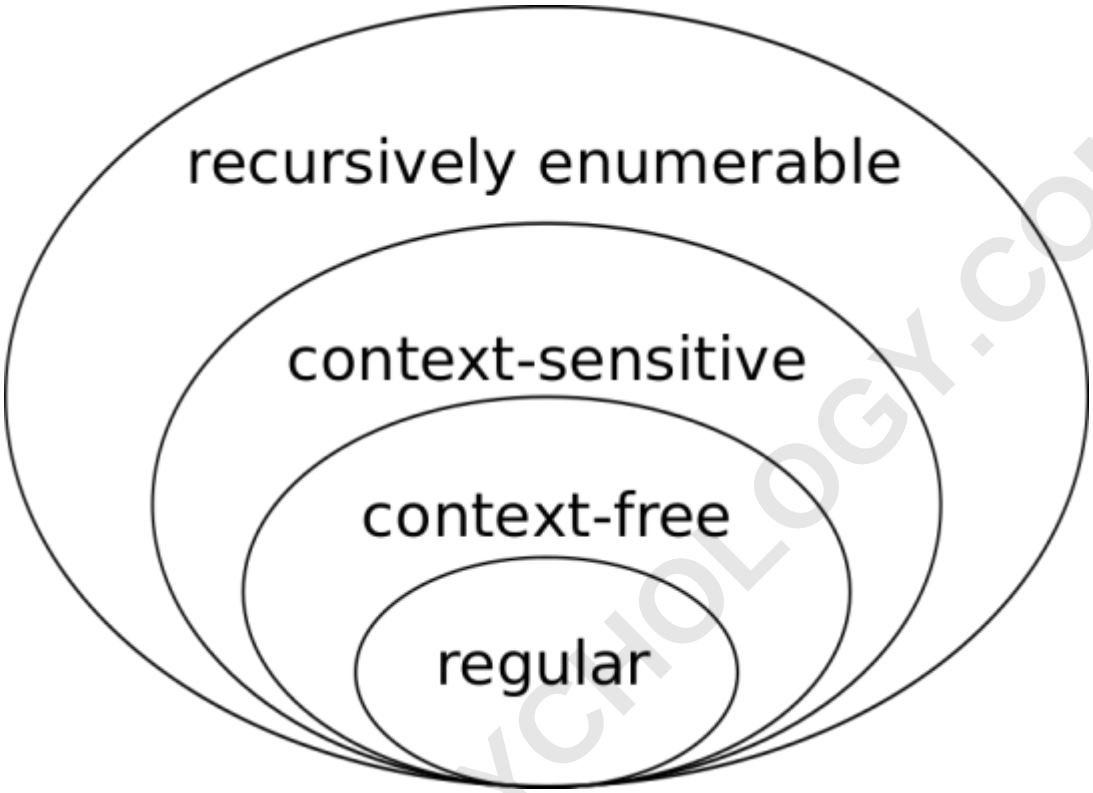
Nonterminals are usually represented by uppercase letters, terminals by lowercase letters, and the start symbol by S. For example, the grammar with terminals {a,b}, nonterminals {S,A,B}, production rules

S -> ABS
S -> ε (where ε is the empty string)
BA -> AB
BS -> b
Bb -> bb
Ab -> ab
Aa -> aa

and start symbol S, defines the language of all words of the form anbn (i.e. n copies of a followed by n copies of b). The following is a simpler grammar that defines the same language: Terminals {a,b}, Nonterminals {S}, Start symbol S, Production rules

S -> aSb
S -> ε

The Chomsky hierarchy.



Set inclusions described by the Chomsky hierarchy.

The Chomsky hierarchy consists of the following levels:

Type-0 grammars (unrestricted grammars) include all formal grammars. They generate exactly all languages that can be recognized by a Turing machine. These languages are also known as the recursively enumerable languages. Note that this is different from the recursive languages which can be decided by an always-halting Turing machine.
Type-1 grammars (context-sensitive grammars) generate the context-sensitive languages. These grammars have rules of the form $\alpha A\beta \rightarrow \alpha\gamma\beta$ with A a nonterminal and α, β and γ strings of terminals and nonterminals. The strings α and β may be empty, but γ must be nonempty. The rule $S \rightarrow \varepsilon$ is allowed if S does not appear on the right side of any rule. The languages described by these grammars are exactly all languages that can be recognized by a linear bounded automaton (a nondeterministic Turing machine whose tape is bounded by a constant times the length of the input.)

Type-2 grammars (context-free grammars) generate the context-free languages. These are defined by rules of the form A -> γ with A a nonterminal and γ a string of terminals and nonterminals. These languages are exactly all languages that can be recognized by a non-deterministic pushdown automaton. Context-free languages are the theoretical basis for the syntax of most programming languages.
Type-3 grammars (regular grammars) generate the regular languages. Such a grammar restricts its rules to a single nonterminal on the left-hand side and a right-hand side consisting of a single terminal, possibly followed (or preceded, but not both in the same grammar) by a single nonterminal. The rule S -> ε is also allowed here if S does not appear on the right side of any rule. These languages are exactly all languages that can be decided by a finite state automaton. Additionally, this family of formal languages can be obtained by regular expressions. Regular languages are commonly used to define search patterns and the lexical structure of programming languages.

Note that the set of grammars corresponding to recursive languages is not a member of this hierarchy. Every regular language is context-free, every context-free language, not containing the empty string, is context-sensitive and every context-sensitive language is recursive and every recursive language is recursively enumerable. These are all proper inclusions, meaning that there exist recursively enumerable languages which are not context-sensitive, context-sensitive languages which are not context-free and context-free languages which are not regular.