Theory of Computation (CS F351)

BITS Pilani
Hyderabad Campus

Dr.R.Gururaj
CS&IS Dept.

CFG Simplification, Classification and Normal Forms

(

CFG Simplification

The definition of context free grammars (CFGs) allows us to develop a wide variety of grammars. Most of the time, some of the productions of CFGs are not useful and are redundant. This happens because the definition of CFGs does not restrict us from making these redundant productions.

By simplifying CFGs we remove all these redundant productions from a grammar , while keeping the transformed grammar equivalent to the original grammar. Two grammars are called equivalent if they produce the same language. Simplifying CFGs is necessary to later convert them into Normal forms.

Simplification steps

We try to Eliminate:

1. NTs not yielding any terminal string
2. NTs not appearing in any sentential form
3. Null productions
4. Unit productions

Source : Theory of Computer Sc Automata Languages and Computation, 3rd Ed., KLP Mishra and N Chandra Sekhar, PHI

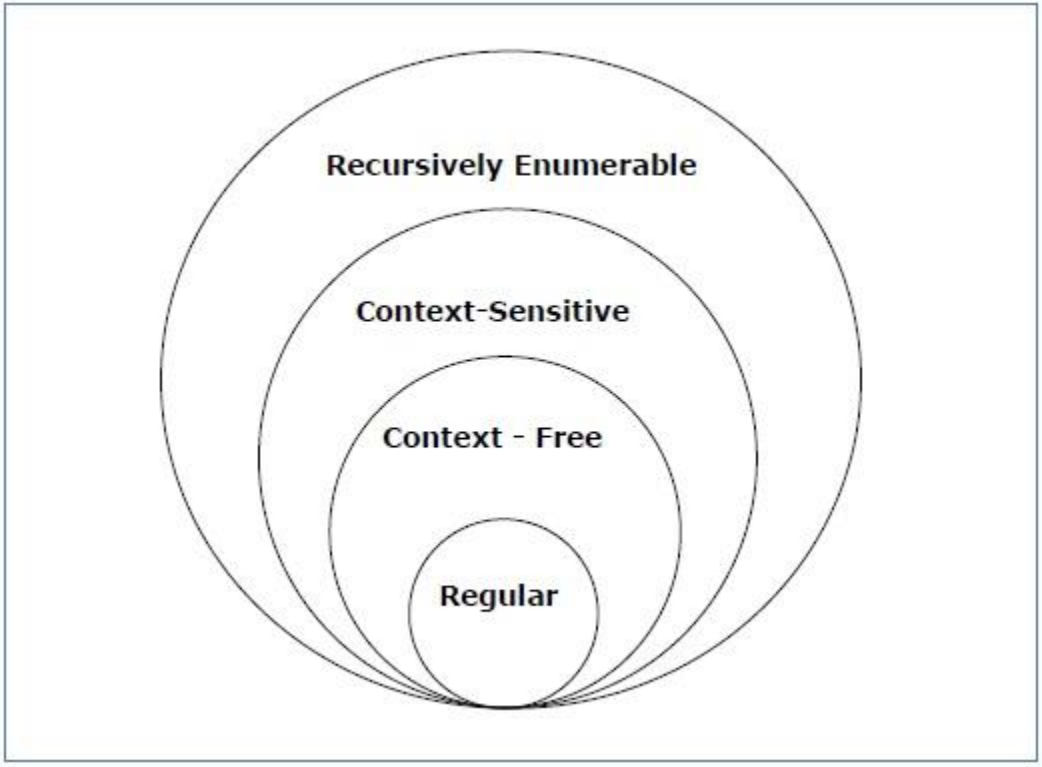
Classification of Grammars

According to Noam Chomosky, there are four types of grammars – Type 0, Type 1, Type 2, and Type 3. The following table shows how they differ from each other –

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	Context-free grammar	Context-free language	Pushdown automaton
Type 3	Regular grammar	Regular language	Finite state automaton

Chomsky Classification of Grammars

The scope of each type of grammar –



Ref: https://www.tutorialspoint.com/automata_theory/chomsky_classification_of_grammars.htm

Type - 3 Grammar

Type-3 grammars generate regular languages. Type-3 grammars must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single non-terminal.

The productions must be in the form **X → a or X → aY**

where **X, Y ∈ N** (Non terminal)

and **a ∈ T** (Terminal)

The rule **S → ε** is allowed if **S** does not appear on the right side of any rule.

X→e; X→ a|aY; Y→b

Type - 2 Grammar

ype-2 grammars generate context-free languages.
The productions must be in the form **A** → **γ**
where **A** ∈ **N** (Non terminal)
and **γ** ∈ **(T** ∪ **N)*** (String of terminals and non-terminals).
These languages generated by these grammars are be recognized by a non-deterministic pushdown automaton.

S → X a
X → a
X → aX
X → abc
X → e

Type - 1 Grammar

innovate achieve lead

Type-1 grammars generate context-sensitive languages. The productions must be in the form **α A β → α γ β**
where **A ∈ N** (Non-terminal)
and **α, β, γ ∈ (T ∪ N)*** (Strings of terminals and non-terminals)
The strings **α** and **β** may be empty, but **γ** must be non-empty.
A→e is not possible.
The rule **S → ε** is allowed if S does not appear on the right side of any rule. The languages generated by these grammars are recognized by a linear bounded automaton.

$AB \rightarrow AbBc$
$A \rightarrow bcA$
$B \rightarrow b$

A linear bounded automaton is like a Turing machine, but with a tape of some bounded finite length.
Length = a function of length of input string.

Type - 0 Grammar

innovate achieve lead

Type-0 grammars generate recursively enumerable languages. The productions have no restrictions. They are any phase structure grammar including all formal grammars. They generate the languages that are recognized by a Turing machine.

The productions can be in the form of **α** → **β** where **α** is a string of terminals and nonterminals with at least one non-terminal and **α** cannot be null. **β** is a string of terminals and non-terminals. A→e is possible.

$S \rightarrow ACaB$

$Bc \rightarrow acB$

$CB \rightarrow DB$

$aD \rightarrow Db$

Normal Forms in CFG

When productions in G satisfy certain restrictions then G is said to be in a '*normal form*'.

There exist several Normal forms. We study-
(1) Chomsky NF,
(2) Greibach NF

When a grammar is in some NF Parser constructions becomes easy.

Chomsky Normal Form (CNF)

The CNF has restriction on length of RHS and nature of symbols in RHS. A CFG G is said to be in CNF

NT→ (singleTerminal) | (NT)(NT))

(1) If every production is of the form A→ a OR
(2) A→BC and
(3) S→ e if e is in L(G)

When *e* is in L(G) we assume that S does not appear on the RHS of any production.

Ex:

(1) S→ AB; S→e ; A→a; B→b is in CNF

(2) S→ABC; S→aC, A→a, B→b, C→c is not in CNF

Conversion into CNF

innovate achieve lead

Steps:

1. Eliminate Null and Unit productions
2. Elimination of Null terminals on RHS
3. Restricting Number of NTs on RHS

Greibach Normal Form (GNF)

The rule is – any production should be of the form

NT→ (singleTerminal) (NT)*

Ex:

(1) S→ aAB; S→b ; A→aA; B→bBC is in GNF

(2) S→ABC; S→aC, A→a, B→b, C→c is not in GNF

Some properties of CFL

CFLs are closed under union, concatenation, and Kleene star.

The intersection of a CFL and RL is a CFL.

CFLs are not closed under intersection and Complementation.