Chomsky and Greibach Normal Forms

Same language.
Standard production forms.

CNF • GNF • Worked conversions

Learning outcomes

Recognize

Identify the allowed production forms in CNF and GNF.

Convert

Apply grammar simplification, terminal isolation and binarization.

Connect

Understand why normal forms support parsing and pushdown automata.

Why normal forms?

Equivalent grammar

Normal forms impose a standard structure while preserving the generated language.

Chomsky Normal Form (CNF)

Binary variable rules make parse trees and dynamic programming easier.

Greibach Normal Form (GNF)

Each nonempty production begins with a terminal; this supports one-symbol-per-step recognition.

Formal definition of CNF

Binary rule

$A \to BC$, where A, B and C are variables.

Terminal rule

$A \to a$, where a is a terminal.

Empty string exception

$S_0 \to \varepsilon$ is permitted if ε belongs to the language and S_0 never appears on a right-hand side.

Recognizing CNF productions

Allowed

$A \to BC$; $B \to a$; $C \to b$. Every nonempty rule is binary or a single terminal.

Not allowed

$A \to aB$; $A \to BCD$; $A \to B$; $A \to \varepsilon$ (except the protected start).

Check

$S_0 \to AB \mid \varepsilon$; $A \to a$; $B \to b$ is CNF and generates $\{ab, \varepsilon\}$.

CNF conversion algorithm

1. Prepare

Add a fresh start $S_0 \to S$. Remove ε-productions, units and useless symbols; preserve ε at S_0 if needed.

2. Isolate terminals

In right-hand sides of length ≥ 2, replace each terminal a with a variable $T_a \to a$.

3. Binarize

Replace each right-hand side of length ≥ 3 with a chain of binary variable rules.

Terminal isolation

Original

$A \to aBb \mid a$. The first rule mixes terminals and variables.

Introduce variables

$T_a \to a$; $T_\beta \to b$.

Replacement

$A \to T_aBT_\beta \mid a$. Keep the single-terminal alternative unchanged; binarize the long alternative next.

Binarization

Original

A → BCD E (four variables).

Introduce fresh variables

X → CD E; then replace X with X → CY and Y → DE.

Binary result

A → BX; X → CY; Y → DE. Expanding X and Y reproduces the original right-hand side.

Solved example 1: prepare

Given grammar

$S \to aSb \mid ab$. Language: $\{a^n b^n : n \geq 1\}$.

New start

$S_0 \to S$; remove the unit rule by copying the alternatives.

After preparation

$S_0 \to aSb \mid ab$; $S \to aSb \mid ab$. No nullable or useless variables remain.

Solved example 1: isolate and split

Terminal variables

$A \to a$; $B \to b$.

Replace mixed terminals

$S_0 \to ASB \mid AB$; $S \to ASB \mid AB$.

Binarize

Introduce $X \to SB$. Final: $S_0 \to AX \mid AB$; $S \to AX \mid AB$; $X \to SB$; $A \to a$; $B \to b$.

Solved example 1: verify

Derive aabb

$S_0 \Rightarrow AX \Rightarrow aX \Rightarrow aSB \Rightarrow aABB \Rightarrow aaBB \Rightarrow aabB \Rightarrow aabb.$

Rule audit

S_0, S and X have only binary variable rules; A and B have single-terminal rules.

Language preservation

AX expands to ASB, corresponding to aSb; AB corresponds to ab.

Solved example 2: preserve ε

Given

S → aSb | ε. Introduce S_0 → S; S is nullable.

Simplify

S_0 → aSb | ab | ε; S → aSb | ab. Remove the original ε-rule and the unit rule.

Final CNF

S_0 → AX | AB | ε; S → AX | AB; X → SB; A → a; B → b. S_0 is absent from every right-hand side.

CNF and parsing

Binary parse trees

A nonempty word of length n has n terminal-producing rules and n−1 binary rules in a CNF derivation.

Rule count

Exactly 2n−1 production applications for any nonempty word of length n.

CYK algorithm

A dynamic programming parser fills a table of variables generating substrings; CNF binary rules combine adjacent substrings.

Greibach Normal Form: overview

Allowed form

$A \to aA_1A_2\ldots A_k$, where a is a terminal, each A_i is a variable and $k \geq 0$.

Example

$S \to aSB \mid aB$; $B \to b$ generates $\{a^n b^n : n \geq 1\}$ and is already in GNF.

Key distinction

CNF starts with two variables or a lone terminal; GNF starts with a terminal followed by zero or more variables. A protected start may generate ε.

GNF conversion: compact example

Given CNF

S → AB; A → a; B → b.

Substitute leading variable

Replace the leading A in S → AB by its terminal-leading production A → a.

Result

S → aB; B → b. Remove unreachable A. General conversion also requires eliminating left recursion and resolving leading-variable dependencies.

GNF conversion: general strategy

Prepare

Remove ε-productions (except protected start), unit productions and useless symbols.

Resolve leading variables

Order variables; substitute leading variables and eliminate left recursion. Continue until each alternative begins with a terminal.

Finish and verify

Replace terminals after the first position by dedicated variables. Check $A \to a\alpha$, where α consists only of variables.

GNF example: eliminate left recursion

Given grammar

S → Sa | b. It generates ba^n for $n \geq 0$.

Rewrite without ε

S → b | bR; R → a | aR. Each step adds one a after the initial b.

Verify

All alternatives start with a terminal and have only variables afterward. S ⇒ bR ⇒ baR ⇒ baa. The language is unchanged.

Exercises and solutions

1. Convert S → abc

Introduce A → a, B → b, C → c and X → BC; replace S → AX.

2. Is S → aB | b; B → b in CNF?

No: aB mixes a terminal and variable. It is in GNF.

3. Recap

Simplify → isolate terminals → binarize. GNF instead ensures terminal-leading alternatives; both preserve context-free languages.