Introduction to Context Free Grammars

From recognition to generation

Finite automata recognize regular languages. Grammars describe how strings are generated.

Core idea

Replace a variable by the right-hand side of a production rule.

Prerequisite

Regular languages, regular expressions and the Pumping Lemma.

Learning outcomes

Define

Identify variables, terminals, production rules and the start symbol.

Construct

Design grammars for simple regular and nonregular languages.

Derive

Generate strings, distinguish sentential forms from sentences, and explain a parse tree.

Why do we need context free grammars?

Regular language

A finite automaton can recognize strings ending in 01.

Beyond regular languages

$L = \{a^n b^n : n \geq 0\}$ is not regular, yet has a simple grammar: $S \to aSb \mid \varepsilon$.

Applications

Programming language syntax, nested parentheses and expression structure.

Formal definition of a CFG

G = (V, Σ, P, S)

V: finite set of variables. Σ: finite set of terminals, disjoint from V.

P: finite set of productions

Each production has the form $A \to \alpha$, where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$.

S ∈ V

The start symbol. ε denotes the empty string; it is not a terminal symbol.

Reading production rules

S → aSb | ε

The vertical bar separates two alternatives: S → aSb and S → ε.

Context free

A variable may be replaced independently of the symbols around it.

Not a CFG production

AB → a is not permitted: the left side must be exactly one variable.

Derivation and generated language

One step: ⇒

If $A \to \alpha$ is a rule, then $uAv \Rightarrow u\alpha v$ for strings u and v of grammar symbols.

Zero or more steps: ⇒*

A sentential form is any α with $S \Rightarrow^* \alpha$; a sentence contains terminals only.

Language of a grammar

$L(G) = \{w \in \Sigma^* : S \Rightarrow^* w\}$. Only finite terminal derivations contribute strings.

Example 1: All binary strings

Grammar

$S \to 0S \mid 1S \mid \varepsilon$

Derive 101

$S \Rightarrow 1S \Rightarrow 10S \Rightarrow 101S \Rightarrow 101$

Language

$L(G) = \{0,1\}^*$. Each step emits one bit; ε terminates the derivation.

Example 2: Equal blocks of a and b

Grammar

$S \to aSb \mid \varepsilon$

Derive aaabbb

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$

Language

$L(G) = \{a^n b^n : n \geq 0\}$. Each recursive step adds one a and one b.

Why the equal-block grammar works

Soundness

After k recursive steps the form is $a^k S b^k$. Replacing S by ε gives $a^k b^k$.

Completeness

For every $n \geq 0$, apply $S \to aSb$ exactly n times, then $S \to \varepsilon$.

Conclusion

The grammar generates exactly the target language, including ε when $n = 0$.

Example 3: Balanced parentheses

Grammar

$S \to (S)S \mid \varepsilon$

Derive ()()

$S \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S)S \Rightarrow ()()S \Rightarrow ()()$

Why it works

The first S generates nested pairs; the second S generates subsequent balanced groups.

Example 4: Binary palindromes

Grammar

$S \to 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$

Derive 0110

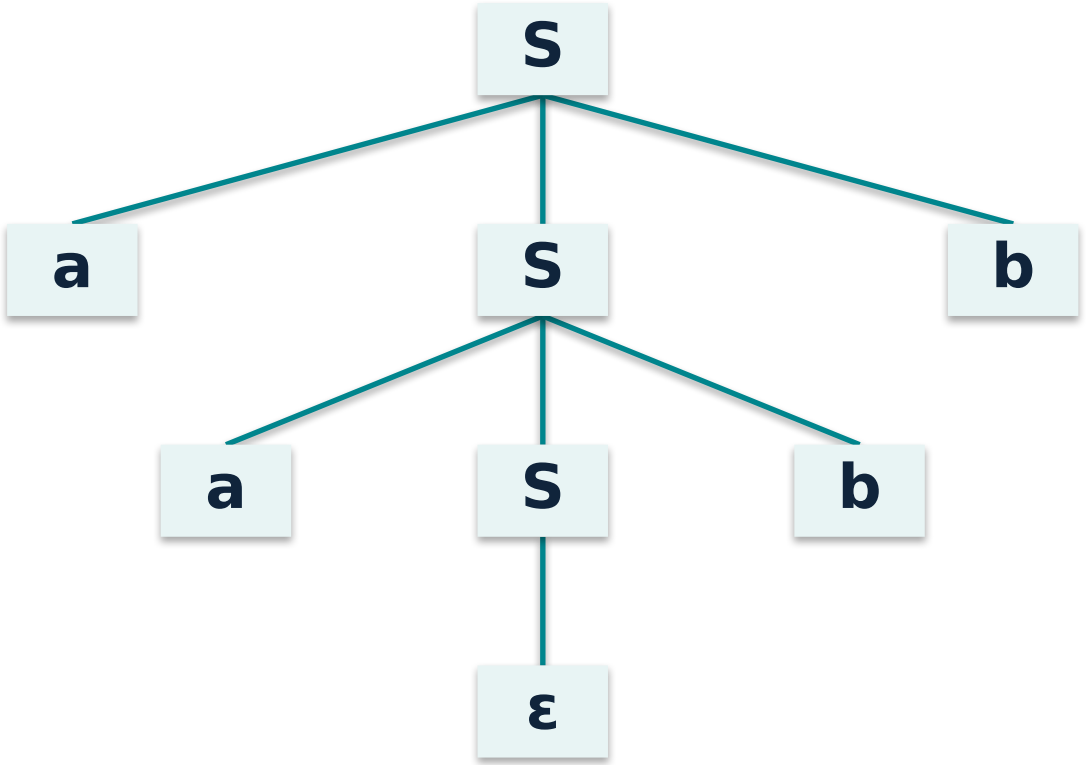
$S \Rightarrow 0S0 \Rightarrow 01S10 \Rightarrow 0110$

Language

All binary palindromes, including ε. Matching outer symbols preserve symmetry.

A first look at parse trees

Grammar: S → aSb | ε • **Yield: aabb**



Read the leaves from left to right, omitting ε.

Leftmost and rightmost derivations

Grammar

$S \to AB,\ A \to aA \mid a,\ B \to bB \mid b$

Leftmost: always expand the leftmost variable

$S \Rightarrow AB \Rightarrow aAB \Rightarrow aaB \Rightarrow aab$

Rightmost: always expand the rightmost variable

$S \Rightarrow AB \Rightarrow Ab \Rightarrow aAb \Rightarrow aab$

Regular languages and CFGs

Every regular language is context free

For a DFA, create variable A_q for each state q; start variable A_{q_0}.

Translate transitions and accepting states

For $\delta(q,a)=r$, add $A_q \to aA_r$. For each accepting q, add $A_q \to \varepsilon$.

Strict inclusion

$\{a^n b^n : n \geq 0\}$ is context free but not regular. Not every language is context free.

Classroom exercises

1. Construct a grammar

Generate all strings $a^n b^m$ where $n,m \geq 0$.

2. Derive a string

Using $S \rightarrow aSb \mid \varepsilon$, derive aabb.

3. Identify a valid CFG rule

Which are valid: $A \rightarrow aB$, $AB \rightarrow a$, $B \rightarrow \varepsilon$? Explain your answer.

Solutions and recap

Exercise 1

S → AB; A → aA | ε; B → bB | ε.

Exercises 2 and 3

S ⇒ aSb ⇒ aaSbb ⇒ aabb. A → aB and B → ε are valid; AB → a is not.

Next lecture

Derivations and Parse Trees: leftmost/rightmost derivations and tree construction.