Derivations and Parse Trees

From production rules to syntactic structure

Leftmost derivation • Rightmost derivation • Parse trees

Learning outcomes

Derive

Apply productions to generate terminal strings.

Distinguish

Compare leftmost and rightmost derivations.

Interpret

Construct parse trees, read their yield and identify distinct structures.

Derivation notation

One step: α ⇒ β

Replace exactly one occurrence of a variable using a production.

Zero or more steps: α ⇒* β

A sequence of zero or more production applications.

Example: S → aSb | ε

S ⇒ aSb ⇒ aaSbb ⇒ aabb. Here aabb is a terminal sentence.

Sentential forms and sentences

Sentential form

Any string of terminals and variables reachable from S: for example, aSb.

Sentence

A sentential form containing only terminals: for example, aabb.

Language

L(G) = { w ∈ Σ* : S ⇒* w }. The empty string ε is also a sentence when derivable.

Leftmost derivation (LMD)

Rule

At every step, replace the leftmost variable in the current sentential form.

Grammar

S → AB; A → aA | a; B → bB | b

Target: aabb

S ⇒ AB ⇒ aAB ⇒ aaB ⇒ aabB ⇒ aabb

Rightmost derivation (RMD)

Rule

At every step, replace the rightmost variable in the current sentential form.

Same grammar

S → AB; A → aA | a; B → bB | b

Target: aabb

S ⇒ AB ⇒ AbB ⇒ Abb ⇒ aAbb ⇒ aabb

Comparing derivation orders

Grammar: S → AB; A → aA | a; B → bB | b

Leftmost	Rightmost
S	S
AB	AB
aAB	AbB
aaB	Abb
aabB	aAbb
aabb	aabb

What is a parse tree?

Root and internal nodes

The root is S. Each expanded variable has ordered children matching a production’s right-hand side.

Leaves

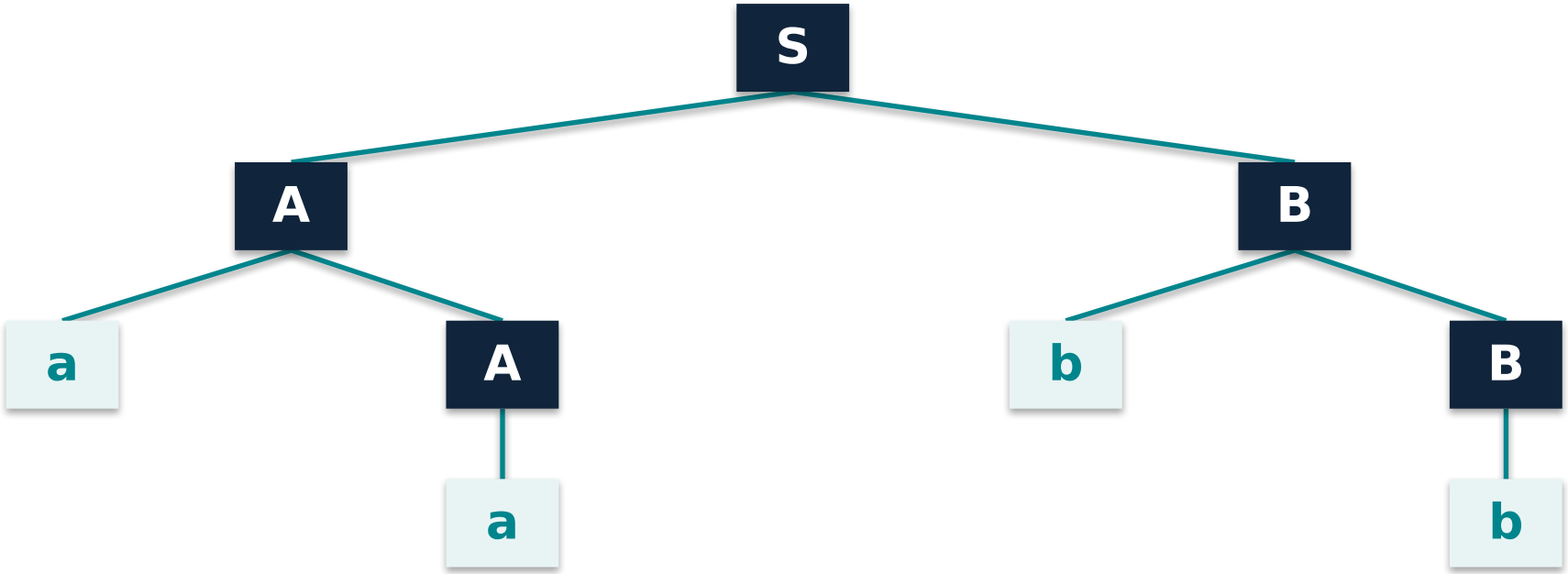
For a complete parse tree, leaves are terminals or ε. Read left to right, ignoring ε.

Key distinction

A parse tree shows structure, not the order in which independent variables were expanded.

Parse tree for aabb

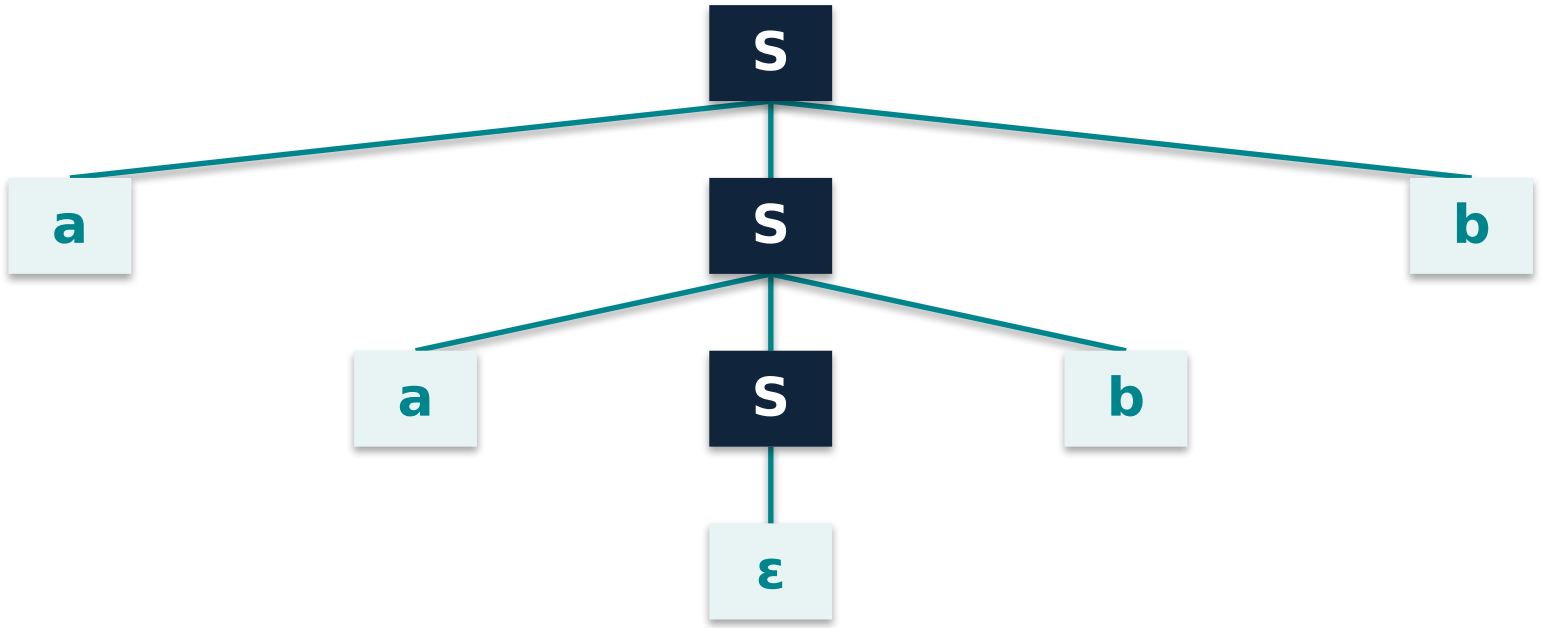
S → AB; A → aA | a; B → bB | b



Yield = a a b b • Both derivation orders produce this tree.

Recursive grammar and ε leaves

S → aSb | ε Target: aabb



S ⇒ aSb ⇒ aaSbb ⇒ aabb | Yield excludes ε.

Recovering derivations from a tree

Leftmost

Expand the leftmost unexpanded variable on the current frontier using its children.

Rightmost

Expand the rightmost unexpanded variable using its children.

Correspondence

Each complete parse tree determines exactly one leftmost and one rightmost derivation.

Expression grammar: precedence

Grammar

$E \to E + T \mid T;\ T \to T * F \mid F;\ F \to id$

Leftmost: id + id * id

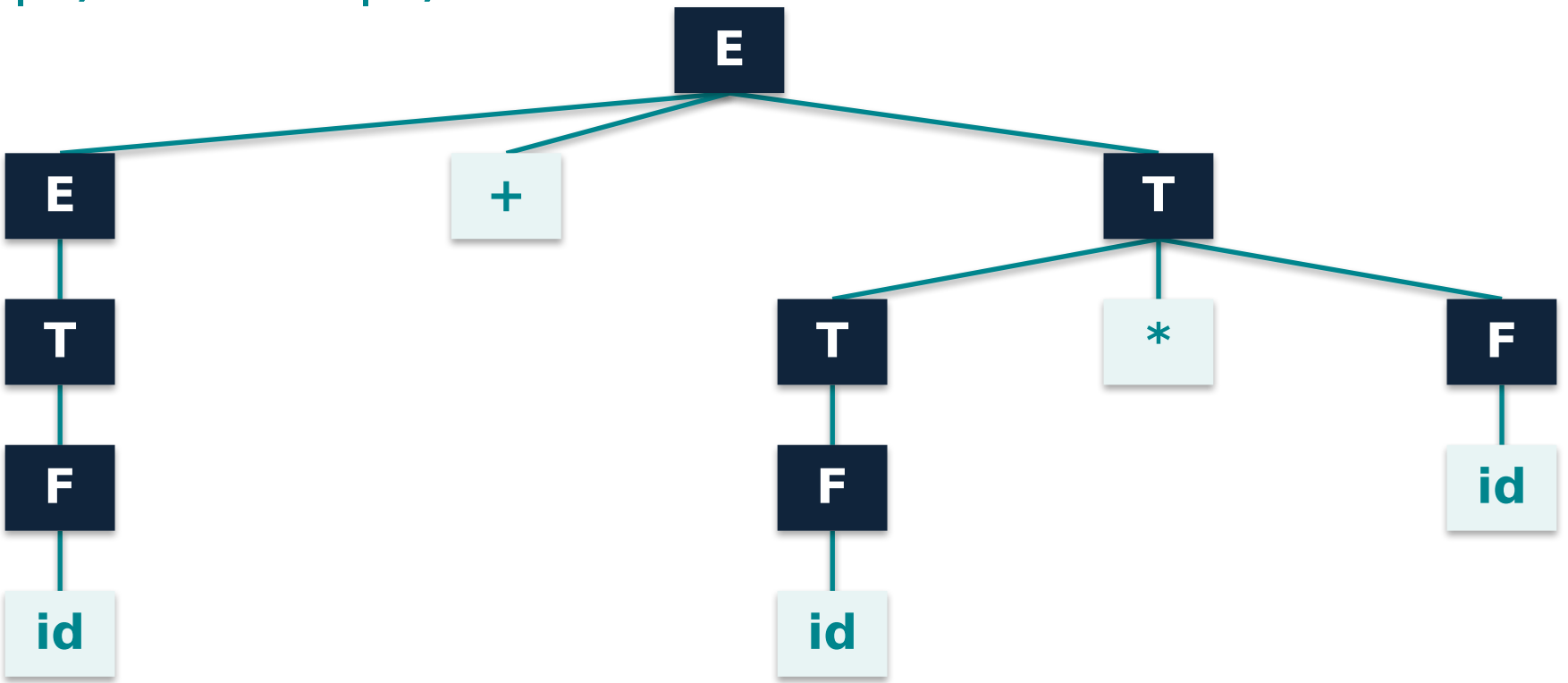
$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow id + T \Rightarrow id + T * F \Rightarrow id + F * F \Rightarrow id + id * F \Rightarrow id + id * id$

Interpretation

Multiplication is nested within T, so it binds more tightly than addition.

Expression parse tree

E → E + T | T; T → T * F | F; F → id



Yield: id + id * id | Structure: id + (id * id)

Ambiguity: a preview

Grammar

$E \to E + E \mid E * E \mid id$

Two distinct structures

The sentence id + id * id can represent (id + id) * id or id + (id * id).

Important

Different expansion orders alone do not prove ambiguity. Two distinct parse trees for one sentence do.

Classroom exercises

Exercise 1

For S → AB; A → aA | a; B → bB | b, give LMD and RMD for aaabb.

Exercise 2

Draw a parse tree for aaabbb using S → aSb | ε.

Exercise 3

For E → E + E | id, give two distinct leftmost derivations of id + id + id.

Solutions and recap

1. Derivation orders

LMD: S ⇒ AB ⇒ aAB ⇒ aaAB ⇒ aaaB ⇒ aaabB ⇒ aaabb

RMD: S ⇒ AB ⇒ AbB ⇒ Abb ⇒ aAbb ⇒ aaAbb ⇒ aaabb

2. Recursive tree

Three nested S → aSb expansions followed by S → ε; yield aaabbb.

3. Two structures

E ⇒ E+E ⇒ E+E+E ⇒ id+E+E ⇒ id+id+E ⇒ id+id+id

E ⇒ E+E ⇒ id+E ⇒ id+E+E ⇒ id+id+E ⇒ id+id+id