

Exercise walk-through

11.2

Context-free languages and parsing theory

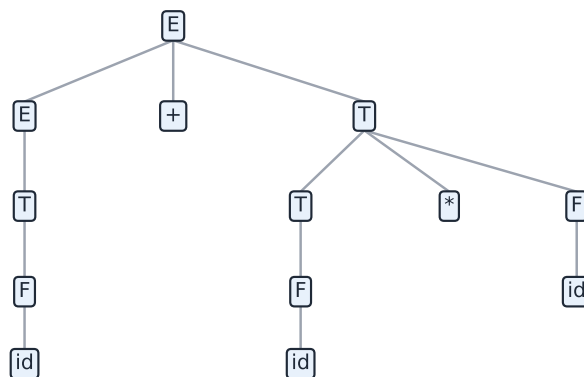
University Computer Science

Context-free: grammar nests

- 1 **Tree** $E \rightarrow E+T, T \rightarrow T*F, F \rightarrow \text{id}$: $\text{id} + \text{id} * \text{id}$ parses once - $*$ binds under T , deeper than $+$.
- 2 **Ambiguity** $E \rightarrow E+E \mid E*E$ | id parses the same string twice - the suite counts 2 trees unlayered, 1 layered.
- 3 **Ceiling** $a^n b^n c^n$ needs two counts: CFGs pump one - context-sensitive territory.

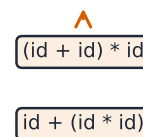
Parse trees

layered: one tree



x binds under T, deeper than + — precedence by layering

unlayered: two trees



same string,
two trees

$E \rightarrow E+E \mid E * E \mid \text{id}$ gives both parses

suite counts: 2 trees unlayered, 1 layered

derivation: $E \Rightarrow E+T \Rightarrow T+T \Rightarrow \text{id}+T \Rightarrow \text{id}+T * F$

$\Rightarrow \text{id}+\text{id} * F \Rightarrow \text{id}+\text{id} * \text{id}$ (six steps)

$a^n b^n c^n$ needs two counts: beyond context-free

Layered: one tree. Unlayered: two trees for the same string.

Exercises 2.1-2.3

2.1 Leftmost derivation of $\text{id} + \text{id} * \text{id}$ under the layered grammar: how many steps? [3]

2.2 What does ambiguity mean, exactly? [3]

2.3 Why does $a^n b^n c^n$ exceed context-free power? [3]

Exercises 2.1-2.3

2.1 Leftmost derivation of $\text{id} + \text{id} * \text{id}$ under the layered grammar: how many steps? [3]

2.1

- Six steps ✓.
-

2.2 What does ambiguity mean, exactly? [3]

2.2

- One string, two distinct parse trees ✓.
-

2.3 Why does $a^n b^n c^n$ exceed context-free power? [3]

2.3

- Two independent counts; the CFG pumping argument handles only one ✓.

Exercise 3.1

3.1 Under the layered grammar, write the full leftmost derivation of $\text{id} * (\text{id} + \text{id})$.

Give each step. State which production each step applies and why that one is forced (lookahead reasoning). Draw or describe the parse tree's shape and count its leaves. State the tree's verdict on the value of $\text{id} * (\text{id} + \text{id})$ versus $(\text{id} * \text{id}) + \text{id}$. [8]

Exercise 3.1

3.1 Under the layered grammar, write the full leftmost derivation of $\text{id} * (\text{id} + \text{id})$.

Give each step. State which production each step applies and why that one is forced (lookahead reasoning). Draw or describe the parse tree's shape and count its leaves. State the tree's verdict on the value of $\text{id} * (\text{id} + \text{id})$ versus $(\text{id} * \text{id}) + \text{id}$. [8]

3.1

- The chain runs $E, T, T \times F, F \times F, \text{id} \times F, \text{id} \times (E), \text{id} \times (E + T), \text{id} \times (T + T), \text{id} \times (\text{id} + T), \text{id} \times (\text{id} + \text{id})$ – nine steps ✓.
- Each step is forced: the leftmost nonterminal with the only production whose FIRST matches the lookahead ✓.
- Tree: $\times$ at the root, left leaf id , right subtree the bracketed $+ - 5$ leaves (**id**, $\times$, **id**, $+$, **id**) ✓.
- Verdict: the product is evaluated first – $\text{id} \times (\text{id} + \text{id})$, not $(\text{id} \times \text{id}) + \text{id}$ ✓.

Exercise 3.2

3.2 The unlayered grammar $E \rightarrow E + E \mid E \times E \mid \text{id}$.

Give both parse trees for `id + id * id` (structure only). Count the trees for `id + id + id` and explain the count. State the danger ambiguity poses to a code generator, and the layered fix with its cost. [7]

Exercise 3.2

3.2 The unlayered grammar $E \rightarrow E + E \mid E \times E \mid \text{id}$.

Give both parse trees for $\text{id} + \text{id} * \text{id}$ (structure only). Count the trees for $\text{id} + \text{id} + \text{id}$ and explain the count. State the danger ambiguity poses to a code generator, and the layered fix with its cost. [7]

3.2

- Tree 1: root $+$, left id , right $\times(\text{id}, \text{id})$. Tree 2: root $\times$, left $+(\text{id}, \text{id})$, right id ✓.
- $\text{id} + \text{id} + \text{id}$: 2 trees (the two association orders) ✓.
- Danger: a code generator must pick one evaluation order – two trees mean two programs ✓.
- Fix: layer ($E/T/F$) so precedence is structural; the cost is a taller tree and re-association for left-recursive idioms ✓.