UNIVERSITY COMPUTER SCIENCE • EXERCISE SHEET

11.2 Context-free languages and parsing theory

Name: _____ Class: _____ Date: _____ **Total: 24 marks**

KEY WORDS parse tree • ambiguity

Objective

Derive with CFGs, count parse trees, and know the next ceiling.

1 Key knowledge and worked examples

■ **The grammar**

$E \to E + T \mid T$; $T \to T \times F \mid F$; $F \to (E) \mid$ **id** ✓. Layering enforces precedence ✓.

■ **Worked example: two trees**

Unlayered $E \to E + E \mid E \times E \mid$ **id** parses `id + id * id` twice (suite-counted); layered parses it once ✓.

2 Practice

2.1 Leftmost derivation of `id + id * 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]

3 Exam-style questions

3.1 Under the layered grammar, write the full leftmost derivation of `id * (id + 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 `id * (id + id)` versus `(id * id) + id`. [8]

3.2 The unlayered grammar $E \to E + E \mid E \times E \mid$ **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]

Solutions

2.1

- Six steps ✓.

2.2

- One string, two distinct parse trees ✓.

2.3

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

3.1

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

3.2

- Tree 1: root +, left id, right ×(id, id). Tree 2: root ×, left +(id, id), right id ✓.
- `id + id + 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 reassociation for left-recursive idioms ✓.

exercises.lol exercises.com.cn