UNIVERSITY COMPUTER SCIENCE • EXERCISE SHEET

11.5 Parsing

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

KEY WORDS context-free languages • pumping lemma

Objective

Run an LL(1) parser by table and stack, and remove left recursion.

1 Key knowledge and worked examples

■ **The machinery**

FIRST/FOLLOW fill the (nonterminal, lookahead) table; the stack runs the derivation; conflicts reject the grammar ✓.

■ **Worked example: the tail trick**

$E \to E + T \mid T$ becomes $E \to TE'$, $E' \to +TE' \mid \varepsilon$ – the loop becomes a tail ✓.

2 Practice

2.1 What fills an LL(1) table cell? [3]

2.2 What does a conflict in a cell mean? [3]

2.3 Why does left recursion break LL(1) parsing? [3]

3 Exam-style questions

3.1 Grammar: $S \to \mathbf{id} = E$; $E \to TE'$; $E' \to +TE' \mid \varepsilon$; $T \to \mathbf{id}$.

Build the LL(1) table (all nonempty cells with justification). Parse `id = id + id`: give each stack snapshot with the lookahead, naming the table cell or the match at each move. State the count of moves and the final verdict. [8]

3.2 Remove left recursion from $A \to A\mathbf{a} \mid A\mathbf{b} \mid \mathbf{c}$.

Give the transformed grammar. Derive `c a b a` under it (leftmost). State what the original grammar generated, in set notation, to show the transformation preserved the language. State the associativity cost and where it is paid. [7]

Solutions

2.1

- The production whose FIRST contains the lookahead (FOLLOW, for an ε production) ✓.

2.2

- Two productions land in one cell: one lookahead cannot choose – the grammar is not LL(1) ✓.

2.3

- The leftmost nonterminal expands to itself: the stack never shrinks and no input is consumed – an infinite loop ✓.

3.1

- Table: on lookahead id, S expands to id = E and E expands to T Ep and T expands to id; on +, Ep expands to + T Ep; at end of input, Ep expands to empty ✓.
- Snapshots, written stack | lookahead: S | id; then id = E | id (apply S); = E | = (match); E | id (match); T Ep | id (apply E); id Ep | id (apply T); Ep | + (match); + T Ep | + (apply Ep); T Ep | id (match); id Ep | id (apply T); Ep | dollar (match; apply Ep to empty) ✓.
- Eleven moves; accept (stack empty at end of input) ✓.

3.2

- Transformed: $A \to \mathbf{c}A'$; $A' \to \mathbf{a}A' \mid \mathbf{b}A' \mid \varepsilon$ ✓.
- Derivation: A, then c A-prime, then c a A-prime, then c a b A-prime, then c a b a A-prime, then c a b a ✓.
- Original language: **c** followed by any string of **a**'s and **b**'s – $\mathbf{c}(\mathbf{a} \mid \mathbf{b})^*$; the transformation generates exactly that ✓.
- Cost: right-recursive trees lean right; left associativity must be restored when building the AST ✓.

exercises.lol exercises.com.cn