Imperial College London
Department of Computing

Compilers (221)

Exercises – LR Parsing

Check your answers with the tutorial helpers during tutorials and with each other on Piazza.

Suggested order to do the questions: 4, 5, 6, 7, 14, 15 and then the remaining questions.

1. Consider the following grammar with start symbol `S`: — L1

```
S → B a | b C
B → d | e B f
C → g C | g S
```

For each of the following strings, give a derivation for the string or say whether the string can or cannot be generated by the grammar:

(i) `da` (ii) `bddf` (iii) `eedffa` (iv) `bggda`

2. Show that all binary strings generated by the following grammar have values divisible by 3. — L3
Hint: use induction on the numerical values for nodes in the parse tree.

```
num → 11 | 1001 | num 0 | num num
```

3. For the rewritten **if** statement grammar in the slides (the grammar with rules for Matched and Unmatched statements on slide 36), draw the parse tree for — L2
if 1 **then** **if** 0 **then** *other* **else** *other*

4. For the following grammar: — L3

```
Statement → begin Statement end | id
```

construct the DFA of LR(0) items and the LR(0) Parsing Table.

You should build the DFA directly without first building the NFA. Your DFA should have 6 states. For conciseness, use the letter `S` in your items instead of `Statement`. Remember to augment the grammar with an auxiliary rule and give numbers for your rules when used in reduce actions in your Parsing Table.

5. Construct the FIRST set and FOLLOW set for the rules (non-terminals) of the following grammar: — L2

```
Statement     → IfStatement | other
IfStatement   → if '(' Expression ')' Statement ElsePart
ElsePart      → else Statement | ε
Expression    → 0 | 1
```

6. Construct the FIRST set and FOLLOW set for the non-terminals of the following grammar: — L2

```
Program       → Statements
Statements    → Statement Statements | ε
Statement     → id '=' Expression | read id | write Expression
Expression    → Term TermTail
TermTail      → AddOp Term TermTail | ε
Term          → Factor FactorTail
FactorTail    → MultOp Factor FactorTail | ε
Factor        → '(' Expression ')' | id | num
AddOp         → '+' | '–'
MultOp        → '*' | '/'
```

7. For the following grammar: — L4

```
Statement → begin Statement end | id
```

construct the DFA of LR(1) items and the LR(1) Parsing Table. Your DFA should have 10 states.

8.	For the following grammar: Statement → **begin** Statement **end** | **id** construct the DFA of LALR(1) items from your solution to the previous question. Your DFA should have 6 states.	L1
9.	For the following grammar: Clock → Clock **tick tock** | **tick tock** i) Construct the DFA of LR(1) items. Use `C` for `Clock`, `i` for **`tick`**, `o` for **`tock`**. Your DFA should have 6 states. ii) Construct the parse table from the DFA of LR(1) items and explain whether the grammar is LR(1).	L3
10.	Consider a robot arm that accepts two commands: **down** that puts an apple in a basket, and **up** that takes an apple out of the basket. Assume the robot arm starts with an empty basket. A valid command sequence for the robot arm should have no prefix that contains more **down** commands than **up** commands, i.e. taking from an empty basket is not permitted. As examples, **down down up up** and **down up down** are valid command sequences, but **up down** and **down up up down** are not. Devise a context-free grammar for all valid command sequences. For your grammar construct the DFA of LR(1) items and explain whether the grammar is LR(1).	L3
11.	Explain which, if any, of SLR(1) and LR(1) can parse the following grammar with start symbol G: G → S | T S → x | z T → y | z	L2
12.	For the following grammar: S → C C C → a C | b ii) Construct the DFA of LR(1) items. ii) Construct the parse table from the DFA of LR(1) items. Your DFA should have 10 states. iii) How many states would the DFA of LALR (1) items have? Explain your answer. iv) Give a regular expression for the strings that `S` recognises.	L4
13.	For the LR(1) example in the slides describe how the input `id = int` would be matched and the AST built by the DFA (slide 30)	L4
14.	Download the calc example from the website and make the parser with flex and bison. On a Mac, download and install Xcode from the App Store. 1) Execute the example with your own expression (warning there may be a command called calc). 2) How many LALR(1) states are did bison generate? Hint: look at the .output file 3) Draw the DFA from the states. 4) Adapt the example to handle division.	L4
15.	Download the extended parser example from the website and make the parser. 1) How many LALR(1) states did bison generate? 2) Which state had the most shift/reduce conflicts? 3) Type in a program consisting of a legal if-then-else statement, i.e. your program should not produce Error: syntax error. Check parser.y for correct syntax.	L2
16.	Download the ANSI C parser example from the website and make the C language parser 1) How many LALR(1) states were generated? 2) Which rule generated a shift-reduce conflict?	L1