https://folk.ntnu.no/michaeng/tdt4205_21/
michael.engel@ntnu.no

Theoretical exercises
Spring 2021

Theoretical Exercises 3

Parsing

Please submit solutions on Blackboard by Friday, 26.02.2021 14:00h

3.1 Top-down parsing: LL(1) form

The following grammar is given:

S $\rightarrow$ a B T | a B T w K
B $\rightarrow$ b
T $\rightarrow$ t | ε
K $\rightarrow$ K s | s

a. Modify the grammar so that it becomes suitable for LL(1) parsing by left factoring and eliminating left recursion.

b. Tabulate the FIRST and FOLLOW sets for all nonterminals of the modified grammar, including which nonterminals are nullable (i.e. they can derive the empty string).

c. Construct the LL(1) parsing table of the modified grammar.

3.2 LR(1) parsing

Consider the grammar (terminal symbols are enclosed in double quotes, for the terminal symbol "Id" the value of the identifier token is a sequence of ASCII characters):

Proto $\rightarrow$ Type "Id" "(" Params ")" ";"
Type $\rightarrow$ "int" | "double" | "char"
Params $\rightarrow$ Params "," Param | Param
Param $\rightarrow$ Type "Id"

a. Trace through the sequence of shift and reduce actions by an LR parser on the input:

```
double blarf(char foo, int bar);
```

Rather than building the configurating sets and LR table and mechanically following the algorithm, try to use your understanding of the shift-reduce technique to guide you.

b. Why does the parser not attempt a reduction to Param after pushing the first sequence of type and identifier onto the parse stack? Doesn't the sequence on top of the stack match the right side? What other requirements must be met before a reduction is performed?

c. How many tokens of lookahead does this parser require? Is the given grammar LR(0) (i.e., it can be parsed by the weakest of the LR parsers) or does it require a larger lookahead?

3.3 LR(1) parse tables

The following grammar is given:

$S \to XX$
$X \to xX \mid y$

Construct the parsing table for this grammar using LR(1).