DEPARTMENT OF COMPUTER SCIENCE, TECHNISCHE UNIVERSITÄT KAISERSLAUTERN
COMPILERS AND LANGUAGE PROCESSING TOOLS - SS17

A Lexer for Mathematical Expressions

Exercise 1
Group 03
Gadler Daniele

27th April 2016

1 OVERVIEW

In the present report, we describe the Lexer we built for parsing and interpreting mathematical expressions. Following, we provide an outline of the main features that it supports. The constructed Lexer passes all default tests provided and other ones written by the report writer.

FEATURES

- **Support for parentheses-enclosed and non-parentheses enclosed expressions**: The resulting Lexer is able to parse and evaluate both parentheses-enclosed expressions (e.g., ((2+2) + (4+2)) as well as non-parentheses-enclosed expressions (e.g:, 1+4 + 2*3). In the latter case, expressions are output with parentheses too (e.g:, 1+4 + 2*3 becomes ((1 + 4) + (2 * 3)) to disambiguate the operations' order.
- **Addition, subtraction, multiplication and division Support**: The resulting Lexer supports the mentioned operations respectively through the following operands: "+", "–", "*", "/", with proper prioritization.
- **Redundant parentheses' support**: The built lexer accepts and evaluates expressions with redundant brackets. (e.g: (((3))) is evaluated to 3 or (((2+1))) is evaluated as (2 + 1).

2 TECHNICAL DESCRIPTION

TASK 2: AST AND BINDING STRENGTH

We solved the binding strength for addition/subtraction and multiplication/division by setting proper priority for operands (from bottom to top), as explained in the official CUP's documentation

[1]. We also set the priority for parentheses, as they need to be parsed into an AST when computing expressions containing redundant brackets (e.g: (((1+2)))).

```
precedence left RPAREN;
precedence left LPAREN;
precedence left  PLUS, MINUS;
precedence left  TIMES, DIV;
```

Particular attention needed to be paid to expressions preceded by a minus sign and negative integers.

- **Negative Expressions**: To distinguish between positive and negative expressions, we implemented an instance attribute named 'sign' in ExprBinary (the parent class of all expressions). If an expression is preceded by a "-", the attribute 'sign' is set to 1. Otherwise, in case of a positive expressions, it is set to 0. In the evaluation, if the attribute 'sign' is 1, the expression will be multiplied by -1.
- **Negative Integers**: After all non-terminal expressions are evaluated, terminal nodes (numbers) are parsed. We distinguish between positive and negative numbers. If a number is preceded by a "-" sign, it is recognized to be negative and it is passed into the new instance of ExprNumber with a '-' in front of it.

ASTs are built by parsing the grammar for all input expressions.

TASK 3 AND 4 - VISITOR PATTERN

We implemented the Visitor Pattern by creating an interface called 'ExprVisitor' that defines methods' signatures for accessing the pretty printing and evaluation functions for the different expressions (e.g., for addition, visitAdd for printing or visitEvaluateAdd for evaluating an expressions). Methods take as input the corresponding expression (e.g: ExprAdd) and return respectively a string for printing or an integer for evaluating the passed expression.
The class 'ExprPrinter' implements the interface 'ExprVisitor' and specifies the 'visit' methods' bodies for the different operations.
In the abstract class 'Expr', 'accept' methods are defined: this class is extended by the class 'ExprBinary', which is again extended by all different expressions' classes (e.g: ExprAdd, ExprSub, ExprMult, ExprDiv) . The only purpose of accept methods is calling the corresponding 'visit' method in the class implementing visit methods (hence, in 'ExprPrinter).
Methods for carrying out expressions (visitOperation) can be thought to work in a 'recursive' manner, recursively accessing the left-hand-side or the right-hand-side of an expression until a number is found. If a number is found, its value is returned and these numbers are processed in the manner defined by the expression itself (e.g: for visitAdd, as an addition).
For further details on the processing of negative numbers, refer to Section 2.

3 BIBLIOGRAPHY

LITERATUR

[1] Official CUP Documentation from TU München. *http://www2.cs.tum.edu/projects/cup/*. Accessed on 26th April 2017.