apl. Prof. Dr. Thomas Noll Philipp Berger, Matthias Volk

Compiler Construction 2018/19

— Exercise Sheet 1 —

Hand in until October 22nd before the exercise class.

General Remarks

- Please hand in your solutions in groups of four members. If you are still looking for a group or your group has less than four members, please post in the L2P forum.
- You may hand in your solutions for the theoretical part of the exercises just before the exercise class starts *or* by dropping them into the "Compiler Construction" box at our chair. Do *not* hand in your solutions for the theoretical part via L2P.
- We do *not* insist on handwritten solutions.
- There also is a practical programming exercise published on a separate exercise sheet.
- We will publish solutions in L2P.
- If you have questions regarding the exercises and/or lecture, feel free to post in the L2P forum, write us an email or visit us at our offices.

Exercise 1 (4 Points)

As an example of the language *WHILE*, consider the following program.

```
1  /* A random walk */
2  int x = 25;
3  int s = 0;
4  while ( x > 0 ) {
5      s++;
6      int b = read() % 10; // randomness by user input
7      if (b <= 4) {
8          x = x - 1;
9      } else {
10         x = x + 1;
11     }
12 }
13 write("I stopped walking after: ");
14 write(s);
15 write(" steps");
```

(a) Give a complete list of the symbol classes and corresponding tokens needed for the lexical analysis of our programming language *WHILE*. Recall that *WHILE* captures (integer) variable declarations, assignments, arithmetic operations, conditional branches, loops, basic I/O (read and write) and Java-style comments.

(b) Decompose the lines 3-6 into a sequence of lexemes and translate each lexeme into a symbol.

Exercise 2 (6 Points)

Let Ω be the set of all relevant characters in some encoding, e.g. all UTF-8 characters. In particular $*, /, \backslash, ", ' \in \Omega$.

(a) Provide a regular expression for single-line comments (`//`) and multi-line comments (`/* ... */`) in *WHILE*. Please keep in mind that `*` as well as `/` may also occur inside of comments. Note that single-line comments are terminated by either a newline symbol `\n` (Linux), a carriage return `\r` (Mac OS) or both `\r\n` (Windows). Your regular expression should support comments on all three operating systems.

Examples of *valid comments* are:

- `// comment\n`
- `/* comment \r new line */`
- `/*foo * bar / */`
- `// /* test */ more comments\r\n`

Examples of *invalid comments* are:

- `// abc`
- `/*two*//*comments*/`

(b) Provide a regular expression capturing a string. Strings begin and end with either double quotation marks (`"`) or single quotation marks (`'`). If a string is enclosed by double quotation marks, then single quotation marks are allowed inside the string (and vice versa). Lastly, an escaped quotation mark (`\"` or `\'`) is also allowed inside a string.

Examples of *valid strings* are:

- `"string"`
- `'02a\a'`
- `"Let's go"`
- `'Let\'s go'`

Examples of invalid strings are:

- `No string`
- `"Two""Strings"`
- `"String`
- `"He said "No""`

(c) Derive *one* NFA that accepts $w \in L_{\text{token}}$, where token is either a comment or a string as defined in the previous tasks.

(d) Solve the simple matching problem on the NFA from (c) for the input `"It\'s"`.

(e) For a regular expression r and natural numbers n, m with $n \leq m$ we define a *repetition operator* $r^{[n,m]}$ to denote n to m occurrences of r. For instance, $L(a^{[1,3]}) = \{a, aa, aaa\}$.
Prove or disprove that for each regular expression with repetition operators, there exists a regular expression without repetition operators.