A Programming Language Concepts

Further Reading

This book does not really explain the compiler construction theory in detail. If you feel that you need to supplement your experience in this area, we recommend Mogensen's textbook for a concise introduction about parsing and lexing (Mogensen, 2011).

If regular expressions are a web-hacking device for you, then we recommend investing in learning about the foundations of this device. This will help you to see which constructs in regular expressions are essential, and which are a syntactic sugar. It will help you to write and read regular expressions more effectively. A good concise introduction to regular expressions from theoretical computer science point of view can be found in Chapter 3.1 of the book by Hopcroft et al. (2001). If you have time, the entire Chapter 3 (relating regular expressions to automata) is highly recommended.

Exercises

The goal of this set of exercises is to recall and "activate" your prior knowledge about compiler construction. You are welcome to skip the exercises that appear trivial—there should be no surprises hidden.

Exercise A.1. Discuss the following questions.

- **a)** Let A and B be arbitrary sets. Does the following equality hold?
  $\mathit{cardinality}(A) + \mathit{cardinality}(B) = \mathit{cardinality}(A \cup B)$
- **b)** Let A and B be arbitrary multisets (bags). Let $\dot\cup$ denote a union of multisets. Is the following equality generally true: $\mathit{cardinality}(A) + \mathit{cardinality}(B) = \mathit{cardinality}(A \dot\cup B)$?
- **c)** Let `A` and `B` be arbitrary finite sequences, and let '.' denote list concatenation. Is the following equality generally true: $\mathit{length}(A) + \mathit{length}(B) = \mathit{length}(A.B)$?

Exercise A.2. Explain in English what are the languages described by the following regular expressions (parentheses are meta-operators used for grouping):

- **a)** $(ab)^*$
- **b)** $1(0|1)^*$
- **c)** $(\ (a(d|c)e)_{\lrcorner})^+$

Examples: The expression ab^* describes the set of all words starting with a symbol a and followed by zero or more bs. The expression $(aa)^+$ describes the language of all non-empty words that can be built from symbol a that have even length.

Exercise A.3. Decide if each of the following strings belongs (or not) to the language generated by the regular expression: `('0'|['0'-'9']+'.'['0'-'9' 'a'-'f']*)`: **a)** `'c0ffee.0730'` **b)** `'0'` **c)** `'1'` **d)** `'0830.c0ffee'` **e)** `'09ea67.'` . Explain why. Notice, that the quotes do not belong to the strings. They are used to denote the places where the strings start and end.

Exercise A.4. Write regular expressions describing the following languages:

a) A language of comma separated tokens (so each word consists of comma separated tokens). Each token consists of one or more occurrences of the letter 'a'. The language contains no other symbols or words.

positive examples	**negative examples**
empty string	a,a,
a	a,b
a,aa	aaa,a,a,a,a,c
aaaaa,a,a,aa	a,aa,aa␣,a

b) A language containing exactly four words; each word is one of the keywords: if, then, else, while

positive examples	**negative examples**
if	if then else
then	if if
else	while if

c) White space consisting of arbitrary many spaces (␣), tab characters(\t), and new line characters (\n) in any order, also mixed.

Exercise A.5. Consider the language L of words such that each word consists of zero or more repetitions of a followed by a single b or a single c. If the final symbol is b the number of as must be even. If the final symbol is c the number of as may be even or odd.

$$L = \{b, aab, aaaab, aaaaaab, \ldots\} \cup \{c, ac, aac, aaac, aaaac, \ldots\}$$

Write a regular expression matching/generating the language L.

Exercise A.6. Consider the following context free grammar (capital letters denote the non-terminals, small letters denote terminals, ϵ denotes empty string, A is the starting symbol):

$$A \rightarrow_1 a\ c\ B\ B \qquad B \rightarrow_3 x\ Bx$$
$$A \rightarrow_2 B\ B\ a\ c \qquad B \rightarrow_4 \epsilon$$

Does the following string belong to the language generated by this grammar: $a\ c\ x\ x\ x\ a\ c$? Argue why not, or show a derivation of the string from the start symbol.

Exercise A.7. Explain in English what is the language described by the following context-free grammar:

$S \rightarrow_1$ $T\ U$ a V $\qquad$ $U \rightarrow_3$ reads | writes

$T \rightarrow_2$ John | Mary | Alice $\qquad$ $V \rightarrow_4$ book | letter | poem

Assume that strings in italics represent non-terminal symbols, and regular text words are terminal symbols. S is the start symbol.

Exercise A.8. Explain in English what language is generated by the following EBNF grammar. Terminal symbols are written in typewriter font, in quotes. Nonterminals are written using capital letters in cursive. S is the start symbol.

$$S \rightarrow_1 S\ OP\ ID \mid ID \qquad OP \rightarrow_2 \texttt{'->'} \mid \texttt{'.'} \qquad ID \rightarrow_3 \texttt{'x'}$$

Exercise A.9. Write a grammar representing the language of balanced parentheses of three kinds, so `"("`, `"{"`, and `"["`, where they can be arbitrarily nested as long as they are always balanced with a closing parenthesis of the same kind. A positive example: `(())[{}]`, a negative example: `([){]}`.

Exercise A.10. Write a simple (possibly) left-recursive grammar in EBNF for an expression language with variables, and conjunction ($\wedge$), disjunction ($\vee$) and negation ($\neg$). Assume that terminals `Id`, `Not`, `And`, and `Or` are defined. They match, in the following order: variable identifiers, negation, conjunction and disjunction operators. Your grammar should be able to generate, among others, the following example expression: $x \wedge \neg(y \vee (z \wedge x \vee \neg y))$

Exercise A.11. Consider the language L of words such that each word consists of zero or more repetitions of a followed by a single b or a single c. If the final symbol is c the number of a's must be even. If the final symbol is b the number of as may be even or odd.

$$L = \{c, aac, aaaac, aaaaaac, \ldots\} \cup \{b, ab, aab, aaab, aaaab, \ldots\}$$

Write a context free grammar in EBNF generating the language L. The grammar may be ambigous and left-recursive (this is fine). Symbols a, b, c will be terminals in your grammar.

Exercise A.12. In the following context-free grammar, capital latin letters denote nonterminals. Small letters and Greek letters denote terminal symbols. Also quoted parentheses are terminals. S is the starting symbol of the grammar.

$$S \rightarrow_1 \alpha\beta S \qquad S \rightarrow_3 {}'('S')' \qquad S \rightarrow_5 d$$
$$S \rightarrow_2 \gamma \qquad S \rightarrow_4 abS$$

Show *two different* derivations, of *different length*, of two different strings from the language generated by this grammar. While you write the derivations, mark the derivation arrows with rule numbers, so that it is easy to reconstruct the order in which the rules are applied.

Exercise A.13. In the following grammar, the start symbol is 'start'. Symbols typeset in typewriter font and strings in quotes are terminals, the other symbols are nonterminals.

start → `'('` parameter-list `')'`
parameter-list → parameter | parameter-list `','`parameter
parameter → `Identifier  Identifier`

Is this grammar left-recursive? If not, explain why below. If yes, eliminate the left-recursion (write down the non-left recursive grammar in EBNF accepting the same language).

Exercise A.14. The following grammar is ambiguous and left-recursive.

$$e \rightarrow e \texttt{ '+' } e \mid \mathbb{Z}$$

The symbols `'+'` and $\mathbb{Z}$ denote terminals; respectively addition and integer numbers.

a) Show an example of a word that can lead to two different parse trees. Draw both trees.
b) Eliminate left-recursion.

Exercise A.15. Consider the following grammar describing a language of simple polynomials

$$P \rightarrow P\ S\ V \texttt{ '\^{}' } N \mid \epsilon \qquad V \rightarrow \texttt{'x'} \mid \texttt{'y'}$$
$$S \rightarrow \texttt{'+'} \mid \texttt{'-'} \qquad N \rightarrow \texttt{'0'} \mid \texttt{'1'} \mid \texttt{'2'}$$

Capital letters denote non-terminals, P is the starting symbol, ϵ is the empty string. Quoted symbols in typewriter font are terminals. Write a regular expression accepting the same language.

Exercise A.16. Consider the following context-free grammar

$$S \rightarrow_1 \texttt{LPAREN}\ L\ \texttt{RPAREN} \qquad L \rightarrow_2 \texttt{ID ','}\ L \qquad L \rightarrow_3 \epsilon$$

where ϵ denotes an empty string of symbols, and the identifiers in typewriter font denote terminals. S and L are non-terminals, and S is the start symbol.

a) Does the string `(a, b, c)` belong to the language generated by this grammar?
b) If yes, please show a derivation, indicating applied rule numbers. If not, fix this grammar so that it belongs to it.
c) Write a regular expression that accepts the same language as accepted by the grammar above.

Exercise A.17. Consider the following context free grammar. In this grammar S is the start symbol, capital letters represent non-terminals, and small letters are terminal symbols. The greek letter ϵ represents an empty string (no characters). Write a regular expression that accepts the same language.

$$S \rightarrow A\ B\ C \qquad B \rightarrow B\ \texttt{b} \mid \epsilon$$
$$A \rightarrow A\ \texttt{a} \mid \epsilon \qquad C \rightarrow C\ \texttt{c} \mid \epsilon$$

Exercise A.18. Draw an abstract syntax tree for the usual parsing of the following expression (guess some reasonable abstract syntax types, or grammar symbols): $x^2 * 2 + \frac{3}{y}$.

Exercise A.19. Draw an abstract syntax tree (guess!) for the following Java expression and annotate nodes with types: `("3" + "4").length()*5` .

Exercise A.20. Consider the following five examples programs. Which part of the compiler will report an error in each of the programs ? (1) lexer/scanner (2) parser (3) name and type analysis? Or is the program statically correct?

a)

```
1 class Main {
2   int main() {
3     return x;
4   }
5 }
```

b)

```
1 class Main {
2   int main() {
3     int x = 0;
4     if (x == 0) {
5       String x = "1";
6       return x;
7     }
8     return 1;
9   }
10 }
```

c)

```
1 class Main {
2   void f() {
3     int 1x = 0;
4   }
5 }
```

d)

```
1 class Main {
2   int main() {
3     int x = 0;
4     if (x == 0) {
5       String y = "1";
6       return x;
7     }
8     return 1;
9   }
10 }
```

e)

```
1 class Main {
2   int main() {
3     if (x == 0) {
4       String y = "1";
5       return 0;
6   }
7 }
```

Exercise A.21. Indicate an expression, a statement, a constant, and a declaration in the program (d) of Exercise A.20.

Exercise A.22. For each of the following error messages from the Java compiler, indicate which compiler part produced the message (lexer/scanner, parser, type checker, runtime)

a) `Line 99: Invalid method declaration; return type required`
b) `Line 33: Variable xx in class Xxxx not accessible from class Yyyy`
c) `Poly.java:4: error: unclosed character literal`
d) `Line 77: '}' expected`

Exercise A.23. What is the value returned by the `run` function in the following Java program:

```
1 class A {
2   public String f (A par) { return "1"; }
3   public String f (B par) { return "2"; }
4 }
5
6 class B extends A {
7
8   public String f (A par) { return "3"; }
9
10  static String run() {
11
12    B x = new B();
13    A y = new B();
14    A z = new A();
15
16    return x.f((A)new B()) + x.f(new B()) + y.f(new A())
17         + y.f((A)new B()) + z.f(new A());
18  }
19 }
```

Exercise A.24. To recall the difference between interpreters and compilers, discuss whether Java (or C#) is an interpreted or a compiled language.