Theoretical Computer Science 1
Exercise Sheet 5

Thomas Haas
Prof. Dr. Roland Meyer

TU Braunschweig
Winter semester 2020/21

Release: 19.01.2021 Due: 29.01.2021, 17:00

Hand in your solutions per E-Mail to your tutor until Friday, 29.01.2021 17:00 pm. You should provide your solutions either directly as .pdf file or as a readable scan/photo of your handwritten notes.

Note: The last task 4. will be content of the lecture next week.

This is the last exercise sheet of the semester. However, all coming lecture content is still relevant for the written exam.

Exercise 1: Closure properties [9 points]

a) [4 points] It is known, that the language $\mathcal{L} = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ is not context-free. Show that its complement $\overline{\mathcal{L}}$, however, is indeed context-free. With this, $\overline{\mathcal{L}}$ is an example of a context-free language whose complement is not context-free.

   Hint: There can be multiple reasons why a word w is not contained in $\mathcal{L}$. Since context-free languages are closed under unions, it is sufficient to consider each of these reasons separately and show that they are context-free.

b) For any word $w = w_1 w_2 \dots w_{n-1} w_n$ we define $\text{reverse}(w) = w_n w_{n-1} \dots w_2 w_1$. For any language $\mathcal{L}$ let $\text{reverse}(\mathcal{L}) = \{\text{reverse}(w) \mid w \in \mathcal{L}\}$.

   - [2 points] Show how to construct from an NFA A a new NFA A' satisfying $\mathcal{L}(A') = \text{reverse}(\mathcal{L}(A))$.
   - [2 points] Now show how to construct a context-free grammar G' from a context-free grammar G such that $\mathcal{L}(G') = \text{reverse}(\mathcal{L}(G))$ holds.
   - [1 point] From the above two points and the fact that right-linear grammars produce regular languages (see previous exercise sheet), deduce that left-linear languages produce regular languages as well.

Exercise 2: CFG, CNF, CYK [8 points]

The CYK algorithm assumes as input a context-free grammar (CFG) in Chomsky normal form (CNF). This means that all production rules are of the form $X \to YZ$ (for non-terminals Y, Z) or of the form $X \to a$ (for a terminal a).

a) [4 points] Use the procedure introduced in the lecture to construct a language-equivalent grammar in CNF for the CFG $G = (\{S, X, Y\}, \{a, b, c\}, P, S)$ defined by the following rules:

$$S \to aXbXc\,,$$
$$X \to Y \mid YYY \mid a\,,$$
$$Y \to bc \mid cb\,.$$

Use your found CNF in conjunction with the Cocke-Younger-Kasami algorithm (CYK algorithm) to decide whether the word *abccbc* is produced by the above grammar G.

b) [4 Punkte] Using the CYK algorithm, decide whether the words *babaa* and *baba* are produced by the following grammar:

$$S \to AB \mid BC,$$
$$A \to CC \mid b,$$
$$B \to BA \mid a,$$
$$C \to AB \mid a.$$

Exercise 3: The syntax of programming languages as grammar [8 points]

In this exercise you will construct a grammar which describes the syntax of a simple programming language.

a) [2 points] Give a context-free grammar G such that its language $\mathcal{L}(G)$ consists of the set of syntactically correct programs as described below.

- Use the terminals id, num, `var`, `if`, `then`, `else`, end, `while`, do, `;`, `+`, `-`, `*`, `/`, `<`, `>`, `=`, `(`, `)`

  "id" is a placeholder for possible variable names and "num" is a placeholder for natural numbers. The other symbols should be self-explanatory.

- An **expression** in the programming language consists of variables, numbers and operations that combine them such as `(x+2)`, `(z<500)`, `(x*(y/3))`, `(x==(y+1))`.

- A **program** is empty, or
  a variable declaration (e.g. `var x;`), or
  an assignment of an expression to a variable (e.g. `x=(x+5);`), or
  a conditional statement (e.g. `if x then y=(z/x); end`), or
  a case distinction (e.g. `if x then y=(z/x); else y=z; end`), or
  a loop (e.g. `while x do x=(x-1); end`), or
  the composition of two programs (e.g. `var x; x=500;`).

b) [2 points] Derive the following program from your grammar in part a) starting from the initial symbol. Give the complete derivation sequence.

```
var x; x=10; var y; y=(x-9); while x do x=(x-1); y=(y+1); end
```

(First you have to replace each variable with id and each number with num.)

c) [2 points] Prove that $\mathcal{L}(G)$ is not regular.

d) [2 points] Describe how to modify the grammar from part a) such that the programming language also supports **functions**.

Functions consist of a name, a parameter list (potentially empty) and a function body. The function body may additionally contain `return;` as well as `return` EXPRESSION`;`. Function calls may be used as statements and as expressions in the program.

For example, the following word should be a valid program:

```
function f (var x) return (x+1); end
function g () var y; y=2; y=f(y); return; end
g();
```

Exercise 4: Pushdown automata [6 points]

Construct pushdown automata for the following languages and state which acceptance condition (empty stack or final states) you assume.

1. [2 points] $\mathcal{L}_1 = \{w \in \{a, b, (,)\}^* \mid w \text{ is correctly parenthesized }\}$.
2. [2 points] $\mathcal{L}_2 = \{w \in \{a, b, (,)\}^* \mid |w|_a = 2|w|_b\}$.
3. [2 points] Can you construct a PDA, which accepts $\mathcal{L}_1 \cap \mathcal{L}_2 = \{w \in \{a, b, (,)\}^* \mid |w|_a = 2|w|_b \text{ and } w \text{ is correctly parenthesized }\}$? If not, what is the intuitive problem here?