Theoretical Computer Science

Winter semester 21/22

Prof. Dr. Georg Schied

Assignment 10

Deadline: Wednesday, 15 December 2021

10 out of 20 points have to be achieved in order to pass.

Exercise 10.1

The following grammar generates sequences of round and square brackets, so that round brackets only can directly contain square brackets and square brackets only directly contain round brackets, e.g.

```
([][])()
[([])]
[][][()([])]
```

S is the start symbol.

$$
\begin{aligned}
S &\rightarrow \quad (\; B \;) \; A \\
&\quad\; | \; [\; A \;] \; B \\
A &\rightarrow \quad (\; B \;) \; A \\
&\quad\; | \; \varepsilon \\
B &\rightarrow \quad [\; A \;] \; B \\
&\quad\; | \; \varepsilon
\end{aligned}
$$

a) Compute *nullable*, *First sets* and *Follow* sets for all non-terminal symbols.

b) Define the predictive parsing table.

c) Is it a LL(1)-grammar?

d) Show step-by-step, how a table driven top-down parser analyzes the input string `[()()]`.

Exercise 10.2 - obligatory (7 points)

The following context-free grammar with start symbol S is given:

$$
\begin{aligned}
S &\rightarrow \quad BSA \;\; | \; cAB \\
A &\rightarrow \quad \varepsilon \quad\;\; | \; bA \\
B &\rightarrow \quad aBAc \; | \; A
\end{aligned}
$$

This grammar has the following properties.

	nullable	First	Follow
S	false	a, b, c	$, b
A	true	b	a, b, c, $
B	true	a, b	a, b, c, $

a) Compute the *predictive parsing table* for this grammar.

b) Is it a *LL(1)-grammar*?

Exercise 10.3 - obligatory (4 points)

The context-free grammar

$S \rightarrow$ c*Sa* | AcB
$A \rightarrow$ a*A* | b
$B \rightarrow$ bB | ε

has the following LL(1) predictive parsing table:

	a	b	c	$
S	S → AcB	S → AcB	S → cSa	-
A	A → aA	A → b	-	-
B	B → ε	B → bB	-	B → ε

a) Show how a table-driven top-down parser analyzes the following input strings. Specify stack content and remaining input for each step.

(1) `aabcb`

(2) `cba`

Which of the strings belong to the language of the grammar?

b) Show a leftmost derivation of the string `aabcb`.

Exercise 10.4 - obligatory (5 points)

The following grammar defines arithmetic expressions, consisting of numbers, operators `+` and `*`, and brackets. As usual, operation `*` has higher precedence than `+`, and both operators are handled as being left-associative.

E → *E* + *T*
| *T*

T → *T* * *F*
| *F*

F → ***number***
| (*E*)

a) Augment the grammar with a power operator `^`, where `3^5` denotes 3^5. The power operator ^ should have higher precedence than multiplication `*` and it should be right-associative, e.g.

```
5*4^3^2
```

should be read as

```
5 * (4 ^ (3 ^ 2)).
```

b) Draw the derivation tree for `5*4^3^2` according to the extended grammar (2, 3, 4, 5 are considered number symbols).

Exercise 10.5

The following grammar is not LL(1). Give three reasons why not (without computing the predictive parsing table).

S → AaS | AB
A → A+A | a
B → Bb | cc

Exercise 10.6

Eliminate left recursion from the following grammar:

S → SAa
| Bb
| Sc

A → aa

B → Bb
| bB
| cc

Exercise 10.7 - obligatory (4 points)

Given is the following excerpt from a grammar for expressions.

Expr → *Expr* + *Term*
| *Expr* − *Term*
| *Term*

a) Give two reasons why the grammar is not LL(1).

b) Transform the productions so that they are no longer left recursive.