Problem Sheet: BNF

Cezar Ionescu

WS 2023-24

1. Develop a grammar for the language of strings consisting of an open bracket `[` followed by a list of zero or more digits separated by commas, followed by a closing bracket `]`.

2. Construct a BNF grammar for the language of parentheses, where the sentences consists of strings of balanced pairs of left and right parens. Examples:

Valid	Invalid
()	())(
(())	(
()()	)(
(((()())()()))	))))))))))(

   Note: the empty string ε should be a sentence in the language.

3. Develop a grammar for simple type declarations in C-like language:

   ```
   int x;
   bool x, y, z;
   static string x;
   ```

4. (a) Develop a grammar that describes football scores in an international tournament, for example:

   ```
   GER-FRA 0:1 (0:0)
   ITA-ENG 3:2 (1:1, 1:1) AE
   SPA-RUS 2:4 (1:1, 2:2) AP
   ```

   `AE` and `AP` are abbreviations for "after extra time" and "after penalties", respectively. The score(s) in parantheses represent the half-time (and regular time) score.

   (b) Extend the above grammar to represent lists of scores. Individual scores are separated by `,` and the list is terminated by a `;` .

5. Extend the grammar of arithmetical expressions to handle exponentiation ^, which has higher priority than + and * and associates to the right (x^y^z `= x^(y^z)`).

6. Consider the following grammar for `if` statements in a simple programming language:

```
   <statement> ::= IF ( <exp> ) <statement> ELSE <statement>
                | IF ( <exp> ) <statement>
 | <other-statement>
```

where <exp> and <`other-statement`> are defined elsewhere in the grammar and represent boolean-valued expressions and other statements (initialization, loops, ...) respectively.

   (a) Show that the grammar of `if` statements is ambiguous.

   (b) Construct an unambiguous grammar (that still allows `if` without an `else`).