Exercise 4

NTNU

TDT4165 fall 2018

Theory solutions

- Formally describe the regular grammar of the lexemes in task 2.

  The lexemes are described by the following regular expressions, which we use to describe regular grammars: You can also draw DFAs/NFAs, but those aren't covered here

  ```
      number: "[0-9]+([0-9]+)?" or simpler "digit+(eps)"
      (if you supported doubles: "[0-9]+(\.[0-9]+)?"  or  "digit+((.digit+) $\vert$ eps)"
      plus: "+"
      minus: "-"
      multiply: "*"
      divide: "/"
      duplicate: "\#"
      additive inverse: "--"
  ```

  You could also have answered using a grammar G = (N, E, P, S) or a similar form, then you would also need to describe each lexeme individually:

  ```
  G_{integer} = ({S,A}, {1,2,3,4,5,6,7,8,9,0}, {
       S -> 1A,
       S -> 2A,
       S -> 3A,
       S -> 4A,
       S -> 5A,
       S -> 6A,
       S -> 7A,
       S -> 8A,
       S -> 9A,
       S -> 0A,

       A -> 1A,
  ```

```
     A -> 2A,
     A -> 3A,
     A -> 4A,
     A -> 5A,
     A -> 6A,
     A -> 7A,
     A -> 8A,
     A -> 9A,
     A -> 0A,
     A -> epsilon,
},
S)

G_{operator} = ({S}, {+,-,*,/,p,d,i}, {
     S -> +,
     S -> -,
     S -> *,
     S -> /,
     S -> p,
     S -> d,
     S -> i,
},
S)
```

- Describe the grammar of the infix notation in task 3 using (E)BNF. Beware of operator precedence. Is the grammar ambiguous? Explain why it is or is not ambiguous?

  Correct answer:

  ```
   Expr ::= Expr + Prod
          | Expr - Prod
          | Prod;
   Prod ::= Prod * number
          | Prod / number
          | number;   Number that was defined as a lexeme
  ```

  This grammar is unambiguous because all parse trees are left-recursive as (((1*2)*3)*4). The grammar parses 1-2-3 into ((1-2)-3), which is semantically correct.

  Partially correct answer (Accepted):

  ```
   Expr ::= Expr + Expr
          | Expr - Expr
          | Prod;
  ```

```
Prod ::= Prod * Prod
       | Prod / Prod
       | number;   Number that was defined as a lexeme
```

This grammar is ambiguous because the parse trees for - say - 1*2*3 can be (1*2)*3 or 1*(2*3). This is wrong because it doesn't capture the precise semantics: 1-2-3 can be (1-(2-3)), which is wrong.

Partially correct answer (Accepted):
Note that you can also create an unambiguous grammar like so: (Wrong answer)

```
Expr ::= Prod + Expr
       | Prod - Expr
       | Prod;
Prod ::= number * Prod
       | number / Prod
       | number;   Number that was defined as a lexeme
```

This grammar is unambiguous because all parse trees are right-recursive as 1*(2*(3*4)). However, the semantics of this grammar are wrong because '-' and '/' are strictly left-associative. This is wrong because it doesn't capture the precise semantics: 1-2-3 is parsed into (1-(2-3)), which is wrong.

- What is the difference between a context-sensitive and a context-free grammar?
  A context-sensitive grammar has a non-terminal surrounded by terminals and/or non-terminals on both the left-hand side and the right-hand side.
  Example: All rules are of the pattern: aAb -¿ aBb
  A context-free grammar does not have this.

- Given the grammar below, determine which of the strings are legal in the language:

```
    <S> ::= <Z> | <X>
    <Z> ::= z <Z> y | z <Y> y | e
    <Y> ::= z <Y> y x | e
    <X> ::= x <X> x | e
```

a) zzyy
b) xxzyxxx
c) xxxx
d) zzyxyx
e) zzzyxyxy

f) zzyxy
g) zxxy

Here you try to process the input through the grammar to see if the matching rules can match the input.

a) zzyy - legal
b) xxzyxxx
c) xxxx - legal
d) zzyxyx
e) zzzyxyxy - legal
f) zzyxy - legal
g) zxxy