Fifth Weekly IPS Assignment

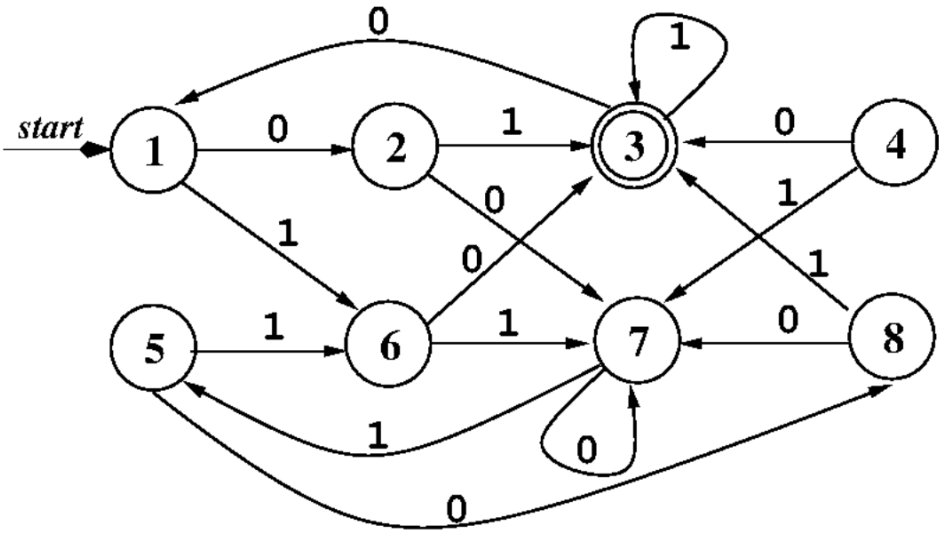
This is the text of the fifth weekly assignment for the DIKU course Implementation of Programming Languages (IPS), 2023.

Hand in your solution in the form of a short report in text or PDF format.

Task 1

This task refers to DFA minimization.

For the alphabet Σ = { `0` , `1` }, minimize the DFA given below, following the algorithm presented in the book and lecture.



Show your derivation steps, not only the final result!

Finally, draw the resulting DFA. (A legible hand-drawn figure is fine.)

Task 2

This task refers to regular expressions and regular languages.

a. Using the alphabet of decimal digits (`0` – `9`), give regular expressions describing the following languages. Also, *briefly* explain how/why your REs work.

- (i) "Proper" numbers (i.e., with no unnecessary leading zeros) that are divisible by 5.
- (ii) Numbers (not necessarily proper) in which digit `5` occurs exactly three times.
- (iii) Numbers satisfying both (i) and (ii).

b. Are the following languages over the alphabet of decimal digits regular? Give short convincing reasons for your answers:

- (i) Numbers (not necessarily proper) that contain digit `1` exactly as many times as digit `2` .
- (ii) Numbers (not necessarily proper) less than 1,000,000, that contain digit `1` exactly as many times as digit `2` .

Task 3

This task refers to how to disambiguate a grammar. See class lecture "Syntax Analysis" slides ~15-20, or Section 2.3 of "Introduction to Compiler Design" book.

The following ambiguous grammar describes boolean expressions:

```
B -> B && B
B -> B || B
B -> ( B )
B -> true
B -> false
```

Rewrite the grammar to be unambiguous, assuming that:

- && and || are single-terminal symbols and that
- && (conjunction) binds tighter than || (disjunction), and that
- && is right associative and || is left associative,

Task 4

This task refers to LL(1) parser construction, including the construction of Nullable, First, Follow sets and eliminating left recursion and identical prefixes. See class lecture "Syntax Analysis" slides ~22-39, or Section 2.8 to 2.12 of "Introduction to Compiler Design" book.

Consider the following grammar for postfix expressions:

```
E -> E E +
E -> E E *
E -> num
```

a. Eliminate Left-Recursion in the grammar (see Slides ~37-38 in class lecture "Syntax Analysis" or Section 2.12.1 in book)

b. Do Left Factorization of the grammar obtained in question (a) (see Slide ~36 in class lecture "Syntax Analysis" or Section 2.12.2 in book)

c. On the grammar obtained in question (b) compute the Nullable and First sets for every production and the Follow sets for every nonterminal (see Slides ~25-31 in class lecture "Syntax Analysis" or Section 2.7-2.10 in book)

d. Compute the Look-Ahead sets for every non-terminal, Make a LL(1) parse table, and Implement a Recursive-Descent Parser (pseudocode). (See Slides ~33-34 and ~41-43 in class lecture "Syntax Analysis" or Section 2.11 in book)

Task 5 (is OPTIONAL)

See class lecture "Syntax Analysis" slides ~45-end, or Section 2.13 to 2.15 of "Introduction to Compiler Design" book.

Consider the grammar below, where ⇒ is considered a single terminal symbol:

```
T   ->   T => T
T   ->   T * T
T   ->   int
```

a. Add a new start production and compute Follow(T) (and remember to add yet another start production to account for the end of file token $).

b. Construct an SLR parser table for the grammar.

c. Eliminate conflicts using the following precedence rules:

   - (i) Operator * binds tighter than ⇒
   - (ii) Operator * is left associative, and ⇒ is right associative.