Computer Language Processing

Exercise Sheet 02

October 7, 2022

Welcome to the second exercise session of CS320! We have selected exercises from the 2019 edition of this class (ex. 1, 2, 3), as well as exercises 3.5, 3.6 from Basics of Compiler Design.

As usual, you can find the solutions of 3.5 and 3.6 on the textbook's website, while solutions of exercises 1, 2, and 3 will be provided to you.

Exercise 1 | Lexical analysis. NFA disjunction

Consider the alphabet A of letter a, b and c. We consider the following lexer:

$$< T_1 : a(ab)^\star, T_2 : b^\star(ac)^\star, T_3 : cba, T_4 : cc^\star >$$

We assume earlier rules for early tokens have priority over later rules.

a) Apply the lexer to the strings below. In each case, show the sequence of tokens output by the lexer:

- $c\,a\,c\,c\,a\,b\,a\,c\,a\,c\,c\,b\,a\,b\,c$
- $c\,c\,c\,a\,a\,b\,a\,b\,a\,c\,c\,b\,a\,b\,c\,c\,b\,a\,b\,a\,c$

b) Convert each of the regular expressions of the lexer to an NFA. Annotate each final state by the token produced by the corresponding rule.

c) Build the disjunction of the NFA from the previous question and convert it to a DFA. Annotate final states by the token with highest priority amongst the annotated tokens.

Exercise 2 | Parallel composition of automatons

In this exercise, we will have a look at the *parallel composition* of deterministic finite state machines.

The *parallel composition* of n automatons is another finite state automaton such that:

- Its state space is the cartesian product of the state spaces of the n automatons.
- Its starting state is the tuple of the starting states of the n automatons.
- Its transition function applies each of the n transitions pointwise.

- Its set of final states is the set of states where at least one of the points is final in its machine.

a) Build a DFA for binary numbers multiples of 2. Do the same for binary numbers multiples of 3.

b) Build the parallel composition of the two automatons from the previous question.

c) Based on your answer from the previous question, show a deterministic finite-state automaton that accepts binary numbers multiples of 2 or 3, **but not both**.

Exercise 3 | Pumping lemma

Let A be the singleton alphabet containing only the symbol 1. Let L be the language of words over A whose size is a prime number.

$$L = \{w \in A^\star \mid |w| \text{ is prime}\}$$

Prove that L is regular by building a regular expression for L, or prove that L is not regular using the pumping lemma. The pumping lemma states that, for any regular language$L$, there exists a strictly positive constant number p, such that every word w in L whose length is at least p can be written as $w = xyz$ where:

1. $|y| > 0$
2. $|xy| \leq p$
3. For any number i, we have that xy^iz is in L.

Exercises from the book

Here are a couple of selected exercises from Basics of Compiler Design to work on grammar.

- **3.5 | Grammar of balanced parentheses**
- **3.6 | An ambiguous grammar**