Syntax and Semantics Exam

Benjamin Bennetzen
Student ID: 20204861
Computer Science, 4th semester

June 7, 2022

1 Exercise 1

1.1

- A $a \cup b \cup c$: is automata ii.
- B Σ^*: automata iv.
- C $a^* \cup b^* \cup c^*$: automata iii.
- D $(a \cup b)^* \cup c^*$: automata i.

1.2

- The union of two regular languages is regular: Yes.
- The union of twe context-free languages is context-free: Yes.
- Assuming the language L is recognized by the NFA $(Q, \Sigma, \delta, q_0, F)$, a possible pumping length for language L would be $p =$: When using the pumping lemma the length of the word has depend on p, however any value bigger than or equal to p will do.

2 Exercise 2

2.1

In state 2 we can notice that that when we read a b from the input it does not depend on what is in the stack, and we also push a b onto the stack. When we read an a from the input we always have to have a b on the stack meaning that at some point before this we should have already read a b. Lastly we should notice that the stack has to be emptied before we can enter the accepting state. This leads me to the following definition for the language $L = \{w \in \Sigma^* \mid |w|_a = |w|_b, |w'|_a \leq |w'|_b\}$, where w' is a prefix of w. This means that in the final word there has to be as many a's as there are b's, but in any given prefix of the word there can be more b's than a's.

2.2

No, the CFG and the PDA do not describe the same language. This can be shown with the word b.

Using the CFG we can derive the word as follows. $S \Rightarrow Sb \Rightarrow \epsilon b$.

However, the PDA will not accapt this word. $1[] \rightarrow_{\epsilon,\epsilon} 2[\$] \rightarrow_{b,\epsilon} 2[b\$]$. We end in a situation where we have a b on the stack, but not more input. Therefore we end in state 2, which is not an accepting state.

3 Exercise 3

Proof by contradiction. We Assume that L is regular. By the pumping lemma we know that there exits a pumping length $p \geq 1$ such that for every $w \in L$, where $|w| \geq p$ then there exits a decomposition $w = xyz$, such that.

1. for all $i \geq 0$, $xy^iz \in L$
2. $|y| > 0$
3. $|xy| \leq p$

We choose the word a^{p^3}, The word is clearly both in L and length at least p. From condition 2 and 3 we know that $y = a^k$ where $1 \leq k \leq p$ When we pump with $i = 2$ we get the following word a^{p^3+k}. Given the constraints on k, $p^3 + k$ cannot be written in the form n^3, where $n = p + 1$. Therefore $a^{p^3+k} \notin L$, thus we cannot satisfy condition 1.

4 Exercise 4

4.1

To prove the two statements i will provide a proof tree for each of them.

First the proof tree for $ab \to 2$

$$\dfrac{\dfrac{}{b \to 2}\text{ by rule 2}}{ab \to 3}\ 3 = 2 + 1, \text{ by rule 3}$$

Then the prooftree for $aab \to 4$.

$$\dfrac{\dfrac{\dfrac{}{b \to 2}\text{ by rule 2}}{ab \to 3}\ 3 = 2 + 1, \text{ by rule 3}}{aab \to 4}\ 4 = 3 + 1, \text{ by rule 3}$$

4.2

First we show thet the property holds for our base case which given that $L' = \{a, b\}^+$, means the words of length 1, those being a and b. From rule 1 we know that $a \to 1$, which upholds the property as $|a|_a + 2|a|_b = 1$. For the word b, we know from rule 2 that $b \to 2$ which again upholds the property as $|b|_a + 2|b|_b = 2$.

For the induction step we have words of length $n+1$. We can divide these words into two cases aw and bw, where w is some word of length n. For the case of aw we have to use rule 3. By our induction hypothesis we know that w upholds the property. For the word aw to uphold the property it the resulting value has to be one bigger than the resulting value of w as there is one more a. In rule 3 that exactly what we do $aw \to k'$, where $k' = k + 1$ and k is the resulting value of w. For the word bw the same arguments can be, but to uphold the property 2 has to be added to the resulting value of w. In rule 4 we can see that $bw \to k'$, where $k' = k + 2$ and k is the resulting value of w, therefore we have that both cases uphold the property.

5 Exercise 5

5.1

With fully static scoping we can make the bindings as seen in Figure 1. With these bindings $y = 3$.

```
begin
   var x:=-1;
   var y:=2;
   proc p is x:=x*y;
   proc q is call p;
   begin
       var x:=3;
       proc p is x:=x-y;
       call q;
       y:=x
   end
end
```

Figure 1: Fully static scope bindings.

5.2

With fully dynamic scoping we can make the bindings as seen in Figure 2. With these bindings $y = 1$.

```
begin
   var x:=-1;
   var y:=2;
   proc p is x:=x*y;
   proc q is call p;
   begin
       var x:=3;
       proc p is x:=x-y;
       call q;
       y:=x
   end
end
```

Figure 2: Fully dynamic scope bindings.

5.3

With the static scoping for procedures and dynamic scoping for variables we can make the bindings as seen in Figure 3. With these bindings $y = 6$.

```
begin
    var x:=-1;
    var y:=2;
    proc p is x:=x*y;
    proc q is call p;
    begin
        var x:=3;
        proc p is x:=x-y;
        call q;
        y:=x
    end
end
```

Figure 3: Mixed scope rules.