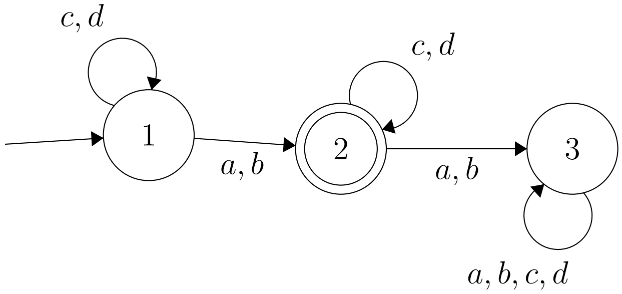
Syntax and Semantics Exam

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

June 4, 2022

1 Exercise

1.1



1.2

$(c|d)^*|((c|d)^*(a|b)(c|d)^*)$

1.3

This is ad-hoc but oh well.

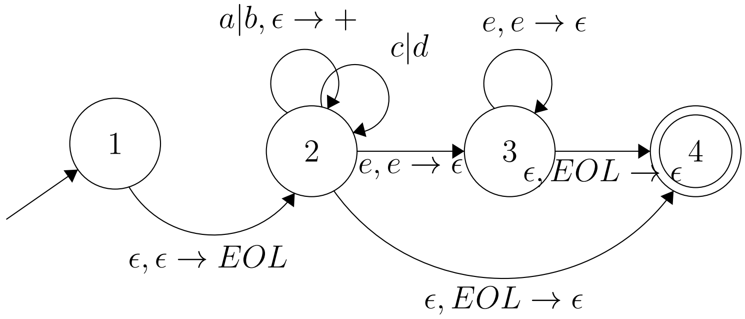
S –> CD AB CD
CD –> c CD | d CD | <empty>
AB –> a | b | <empty>

2 Exercise

2.1

```
S -> a S e
   | b S e
   | c S
   | d S
   | <empty>
```

2.2



$a|b, \epsilon \to +$
$c|d$
$e, e \to \epsilon$
1
2
3
4
$e, e \to \epsilon$
$\epsilon, EOL \to \epsilon$
$\epsilon, \epsilon \to EOL$
$\epsilon, EOL \to \epsilon$

$1, 2, 2, 2, 3, 3, 4$

3 Exercise

We choose the word $a^p e^p \in L'$ which is clearly of longer than p. Since $|xy| \leq p$, $z = e^p$ and $y = a^q$, where $1 \leq q \leq p$, as $|y| > 0$. This means that $xy^0 z = a^{p-q} e^p \notin L'$.

4 Exercise

4.1

$$\begin{aligned}
s(s(s(0))) + s(s(0)) &\Rightarrow s(s(0)) + s(s(s(0))) && [r_3]\\
&\Rightarrow s(0) + s(s(s(s(0)))) && [r_1]\\
&\Rightarrow 0 + s(s(s(s(s(0))))) && [r_1]\\
&\Rightarrow s(s(s(s(s(0))))) && [r_2]
\end{aligned}$$

4.2

$$[r_5] \quad a_1 * a_2 \Rightarrow a_2 * a_1 \tag{1}$$

$$[r_6] \quad 0 * a_1 \Rightarrow 0 \tag{2}$$

$$[r_7] \quad s(0) * a_1 \Rightarrow a_1 \tag{3}$$

$$[r_8] \quad s(n_1) * a_1 \Rightarrow n_1 * a_1 + a_1 \tag{4}$$

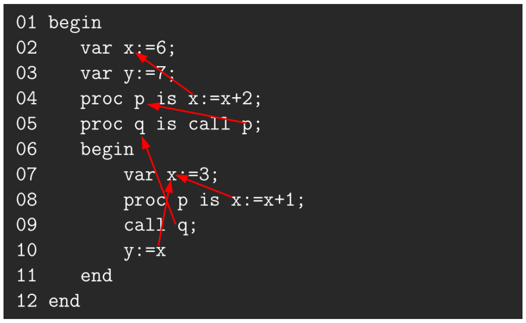
4.3

$$\begin{aligned}
s(s(0)) * s(s(0)) &\Rightarrow s(0) * s(s(0)) + s(s(0)) && [r_8]\\
&\Rightarrow s(s(0)) + s(s(0)) && [r_7]\\
&\Rightarrow s(0) + s(s(s(0))) && [r_1]\\
&\Rightarrow 0 + s(s(s(s(0)))) && [r_1]\\
&\Rightarrow s(s(s(s(0)))) && [r_2]
\end{aligned}$$

5 Exercise

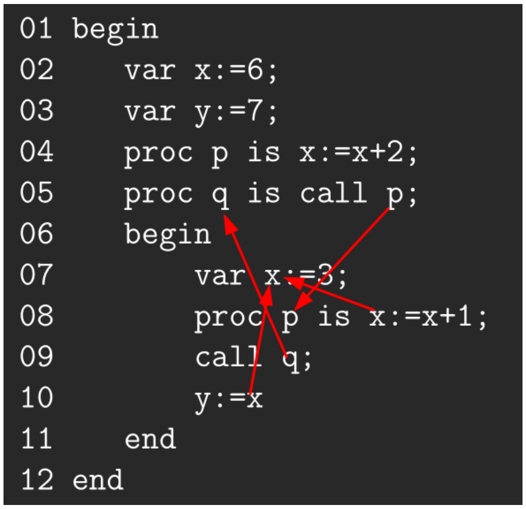
5.1

When declaring a procedure using fully static scope rules we save both the current variables environment and procedure environment. And these are retrieved when calling a procedure. Thus we can make the following bindings as seen in Figure 1, and figure out that $y = 3$.



```
begin
    var x:=6;
    var y:=7;
    proc p is x:=x+2;
    proc q is call p;
    begin
        var x:=3;
        proc p is x:=x+1;
        call q;
        y:=x
    end
end
```

Figure 1: fully static scope bindings.



```
begin
    var x:=6;
    var y:=7;
    proc p is x:=x+2;
    proc q is call p;
    begin
        var x:=3;
        proc p is x:=x+1;
        call q;
        y:=x
    end
end
```

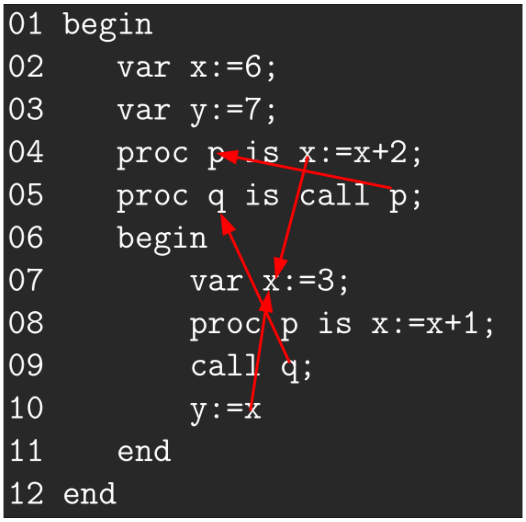
Figure 2: fully dynamic scoping.

5.2

In a fully dynamic scope we only save the procedure when we declare, and in the call of the procedure we simply use the variables and procedures in their respective current environments. Again we can draw the bindings as can be seen on Figure 2. Thus in this case $y = 4$

5.3

With these scope rules we will save the procedure environment when declaring a procedure, but use the current variable environment when calling one. On



```
begin
    var x:=6;
    var y:=7;
    proc p is x:=x+2;
    proc q is call p;
    begin
        var x:=3;
        proc p is x:=x+1;
        call q;
        y:=x
    end
end
```

Figure 3: Mixed scoping rules.

Figure 3 i have again shown the bindings. This time $y = 5$

6 Introduction

This is time for all good men to come to the aid of their party!

$$\dfrac{\dfrac{}{\mathcal{N}[\![\underline{2}]\!] \to 2}\text{ by num} \qquad \dfrac{}{\mathcal{N}[\![\underline{2}]\!] \to 2}\text{ by num}}{\underline{2} + \underline{2} \to 4}\text{ by Plus}$$