

## Übung 4, Aufgabe 3 a

```
let val x = 0
in ( $\lambda x:\text{bool}.$  if  $x$  then 1 else  $\sim 1$ ) ( $x \geq 0$ )
end
```

steht für

```
let val x = 0
in ( $\lambda x:\text{bool}.$  if  $x$  then 1 else  $\sim 1$ ) ( $\geq (x, 0)$ )
end
```

→ ( $\lambda x:\text{bool}.$  if  $x$  then 1 else  $\sim 1$ ) ( $\geq (0, 0)$ )  
mit Regel (LET-EXEC)

→ ( $\lambda x:\text{bool}.$  if  $x$  then 1 else  $\sim 1$ ) *true*  
mit Regel (OP)

→ if *true* then 1 else  $\sim 1$   
mit Regel (BETA-V)

→ 1  
mit Regel (COND-TRUE)

## Übung 4, Aufgabe 3 d

```
let val f =  $\lambda x:\text{int}.$  0
    val f =  $\lambda x:\text{int}.$  if  $x = 0$  then 1 else  $f (x - 1)$ 
in f 3
end
```

steht für

```
let val f =  $\lambda x:\text{int}.$  0
in let val f =  $\lambda x:\text{int}.$  if  $= (x, 0)$  then 1 else  $f (- (x, 1))$ 
    in f 3
    end
end
```

→ let val  $f = \lambda x:\text{int}.$  if  $= (x, 0)$  then 1 else ( $\lambda x:\text{int}.$  0) ( $- (x, 1)$ )  
in f 3  
end

mit Regel (LET-EXEC)

→ ( $\lambda x:\text{int}.$  if  $= (x, 0)$  then 1 else ( $\lambda x:\text{int}.$  0) ( $- (x, 1)$ )) 3  
mit Regel (LET-EXEC)

→ if  $= (3, 0)$  then 1 else ( $\lambda x:\text{int}.$  0) ( $- (3, 1)$ )  
mit Regel (BETA-V)

→ if *false* then 1 else ( $\lambda x:\text{int}.$  0) ( $- (3, 1)$ )  
mit Regel (OP)

→ ( $\lambda x:\text{int}.$  0) ( $- (3, 1)$ )  
mit Regel (COND-FALSE)

→ ( $\lambda x:\text{int}.$  0) 2  
mit Regel (OP)

→ 0  
mit Regel (BETA-V)

## Übung 4, Aufgabe 3 b

```
let val x = 0
    val x = x = x
in x
end
```

steht für

```
let val x = 0
in let val x == (x, x)
    in x
    end
end
```

→ let val x == (0, 0)  
 in x  
 end

mit Regel (LET-EXEC)

→ let val x = true  
 in x  
 end

mit Regel (OP)

→ true  
 mit Regel (LET-EXEC)

## Übung 4, Aufgabe 3 c

```
let val f = λx:int. if x = 0 then 1 else f (x - 1)
in f 3
end
```

steht für

```
let val f = λx:int. if = (x, 0) then 1 else f (- (x, 1))
in f 3
end
```

→ (λx:int. if = (x, 0) then 1 else f (- (x, 1))) 3  
 mit Regel (LET-EXEC)

→ if = (3, 0) then 1 else f (- (3, 1))  
 mit Regel (BETA-V)

→ if false then 1 else f (- (3, 1))  
 mit Regel (OP)

→ f (- (3, 1))  
 mit Regel (COND-FALSE)

→ f 2  
 mit Regel (OP)

Das Programm bleibt hier stecken wegen des frei vorkommenden Namens *f*.

## Übung 4, Aufgabe 3 e

```
let val x = 0
    val f = λ y:bool. x
    val x = true
in f x
end
```

steht für

```
let val x = 0
in let val f = λ y:bool. x
    in let val x = true
        in f x
        end
    end
end
```

→ let val f = λ y:bool. 0  
in let val x = true  
in f x  
end  
end

mit Regel (LET-EXEC)

→ let val x = true  
in (λ y:bool. 0) x  
end

mit Regel (LET-EXEC)

→ (λ y:bool. 0) true  
mit Regel (LET-EXEC)

→ 0  
mit Regel (BETA-V)

## Übung 4, Aufgabe 3 f

```
let val g = λx:int. 0
    val f = λx:int. g x
    val g = λx:int. true
in f 0
end
```

steht für

```
let val g = λx:int. 0
in let val f = λx:int. g x
    in let val g = λx:int. true
        in f 0
        end
    end
end
```

→ let val f = λx:int. (λx:int. 0) x  
in let val g = λx:int. true  
in f 0  
end  
end

mit Regel (LET-EXEC)

→ let val g = λx:int. true  
in (λx:int. (λx:int. 0) x) 0  
end

mit Regel (LET-EXEC)

→ (λx:int. (λx:int. 0) x) 0  
mit Regel (LET-EXEC)

→ (λx:int. 0) 0  
mit Regel (BETA-V)

→ 0  
mit Regel (BETA-V)