

Small step Semantik

Berechnung des ggT mit dem Euklidischen Algorithmus

```
let fun gcd (m:int) (n:int) : int =  
    if n = 0 then m else gcd n (m mod n)  
in gcd 10 4  
end
```

steht für

```
let val gcd = rec gcd : int → int → int.  
    λ m : int. λ n : int. if = (n, 0) then m else gcd n (mod (m, n))  
in gcd 10 4  
end
```

→ let val gcd = λ m : int. λ n : int. if = (n, 0) then m else (rec gcd : int → int → int. ...) n (mod (m, n))
in gcd 10 4
end

mit Regel (UNFOLD)

→ (λ m : int. λ n : int. if = (n, 0) then m else (rec gcd : int → int → int. ...) n (mod (m, n))) 10 4
mit Regel (LET-EXEC)

→ (λ n : int. if = (n, 0) then 10 else (rec gcd : int → int → int. ...) n (mod (10, n))) 4
mit Regel (BETA-V)

→ if = (4, 0) then 10 else (rec gcd : int → int → int. ...) 4 (mod (10, 4))
mit Regel (BETA-V)

→ if false then 10 else (rec gcd : int → int → int. ...) 4 (mod (10, 4))
mit Regel (OP)

→ (rec gcd : int → int → int. ...) 4 (mod (10, 4))
mit Regel (COND-FALSE)

→ (λ m : int. λ n : int. if = (n, 0) then m else (rec gcd : int → int → int. ...) n (mod (m, n))) 4 (mod (10, 4))
mit Regel (UNFOLD)

→ (λ n : int. if = (n, 0) then 4 else (rec gcd : int → int → int. ...) n (mod (4, n))) (mod (10, 4))
mit Regel (BETA-V)

→ (λ n : int. if = (n, 0) then 4 else (rec gcd : int → int → int. ...) n (mod (4, n))) 2
mit Regel (OP)

→ if = (2, 0) then 4 else (rec gcd : int → int → int. ...) 2 (mod (4, 2))
mit Regel (BETA-V)

→ if false then 4 else (rec gcd : int → int → int. ...) 2 (mod (4, 2))
mit Regel (OP)

→ (rec gcd : int → int → int. ...) 2 (mod (4, 2))
mit Regel (COND-FALSE)

→ (λ m : int. λ n : int. if = (n, 0) then m else (rec gcd : int → int → int. ...) n (mod (m, n))) 2 (mod (4, 2))
mit Regel (UNFOLD)

→ (λ n : int. if = (n, 0) then 2 else (rec gcd : int → int → int. ...) n (mod (2, n))) (mod (4, 2))
mit Regel (BETA-V)

→ (λ n : int. if = (n, 0) then 2 else (rec gcd : int → int → int. ...) n (mod (2, n))) 0
mit Regel (OP)

→ if = (0, 0) then 2 else (rec gcd : int → int → int. ...) 0 (mod (2, 0))
mit Regel (BETA-V)

→ if true then 2 else (rec gcd : int → int → int. ...) 0 (mod (2, 0))
mit Regel (OP)

→ 2
mit Regel (COND-TRUE)