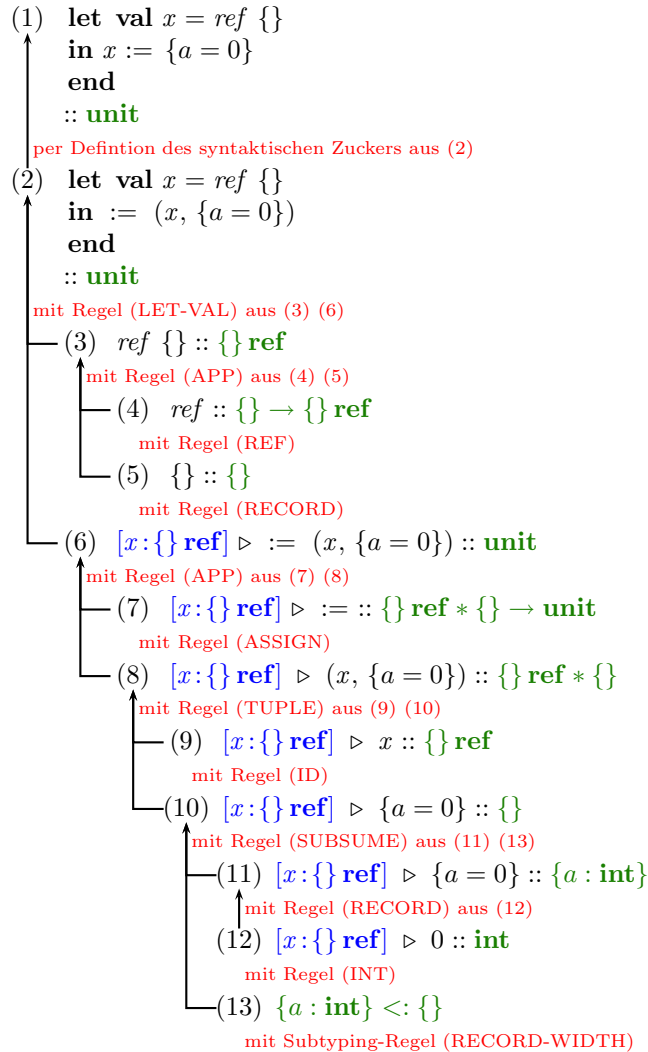


## Übungsblatt 12, Aufgabe 1 a

Das Programm ist wohlgetypt und es terminiert.

### Typüberprüfung



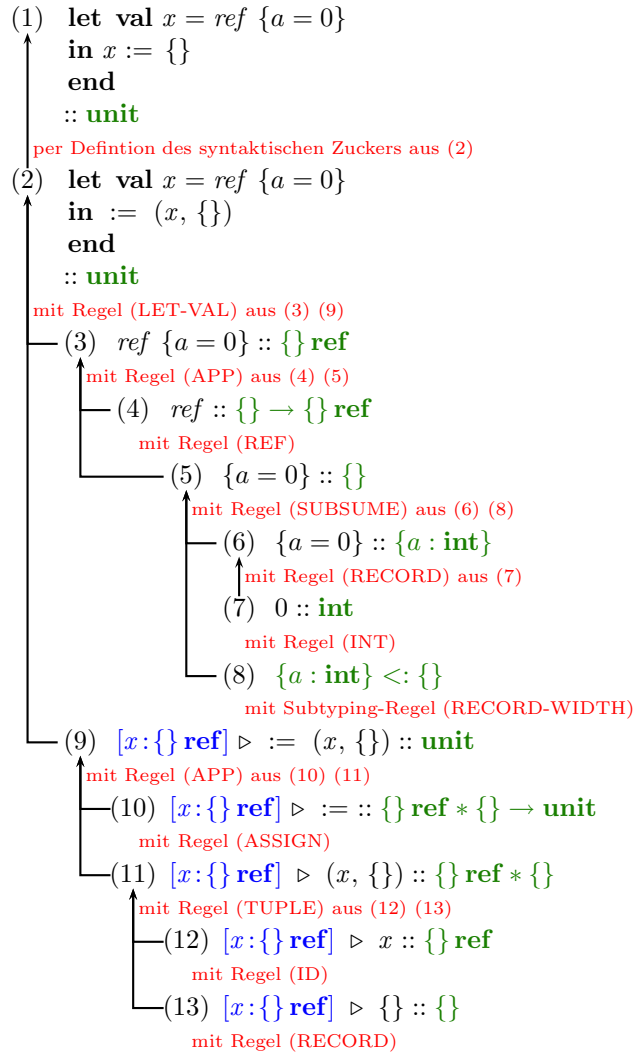
### Small step Semantik

(**let val**  $x = \text{ref } \{\}$  **in**  $:= (x, \{a = 0\})$  **end**,  $[\ ]$ )  
 $\rightarrow$  (**let val**  $x = X$  **in**  $:= (x, \{a = 0\})$  **end**,  $[X : \{\}]$ )  
 mit Regel (REF)  
 $\rightarrow$  ( $:= (X, \{a = 0\})$ ,  $[X : \{\}]$ )  
 mit Regel (LET-EXEC)  
 $\rightarrow$  ( $()$ ,  $[X : \{a = 0\}]$ )  
 mit Regel (ASSIGN)

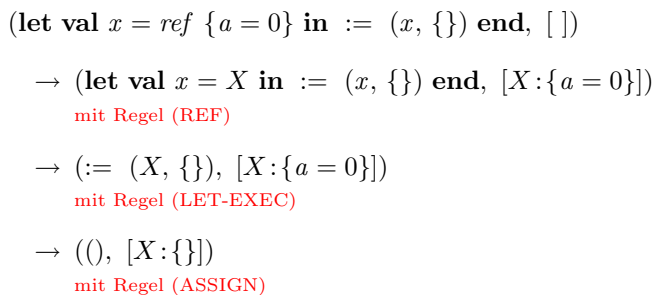
## Übungsblatt 12, Aufgabe 1 b

Das Programm ist wohlgetypt und es terminiert.

### Typüberprüfung



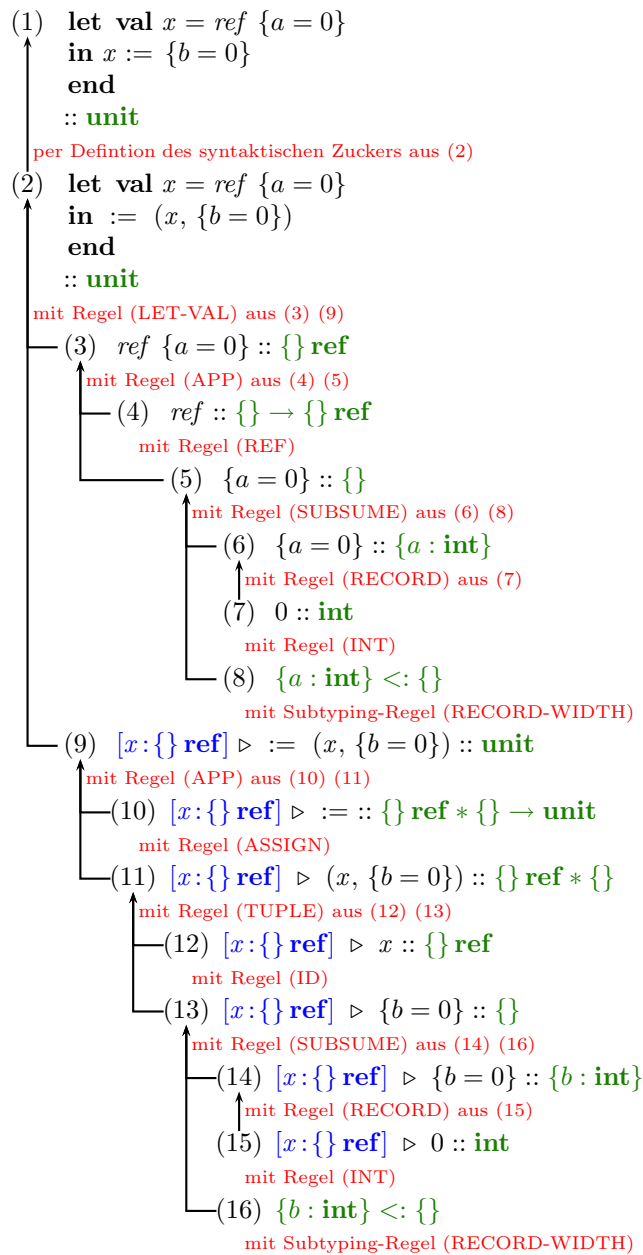
### Small step Semantik



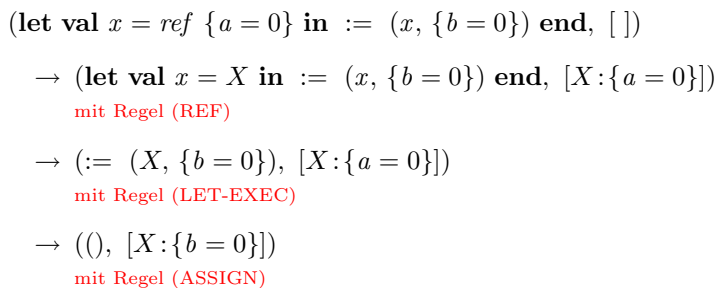
## Übungsblatt 12, Aufgabe 1 c

Das Programm ist wohlgetypt und es terminiert.

### Typüberprüfung



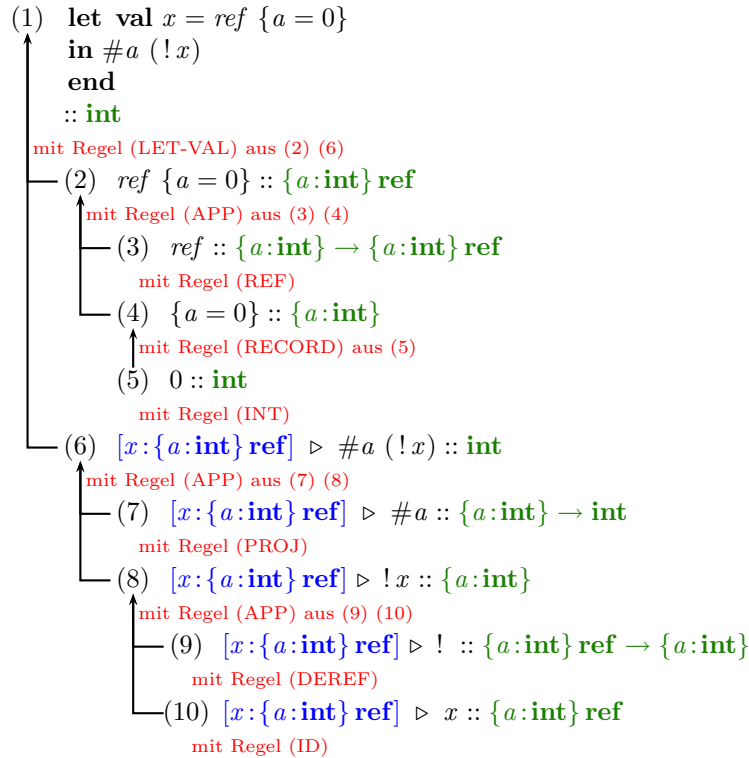
### Small step Semantik



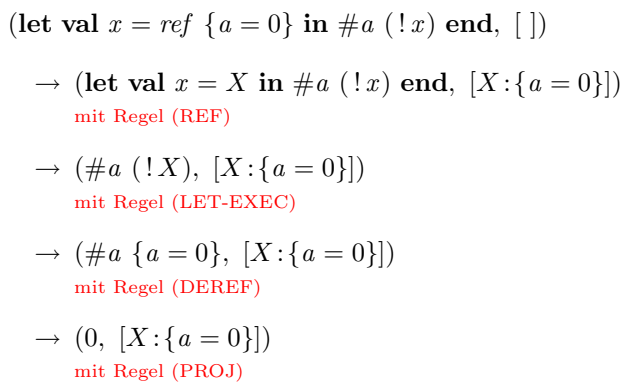
## Übungsblatt 12, Aufgabe 1 d

Das Programm ist wohlgetypt (ohne Subtyping-Regeln) und es terminiert.

### Typüberprüfung



### Small step Semantik



## Übungsblatt 12, Aufgabe 1 e

Das Programm ist *nicht* wohlgetypt und es bleibt stecken.

### Typüberprüfung

Bei der Deklaration muss man sich entscheiden, ob man für  $x$  den Typ  $\{\}$  **ref** oder den Typ  $\{a : \mathbf{int}\}$  **ref** wählt. Im ersten Fall ist  $x := \{\}$  wohlgetypt (vgl. Aufgabe 1 b), aber  $\#a (!x)$  nicht. Im zweiten Fall scheitert man schon an der Zuweisung, da  $x : \{a : \mathbf{int}\}$  **ref** keinen anderen Typ mehr annehmen kann (Subtyping-Regel (REF)) und  $\{\}$  nicht vom Typ  $\{a : \mathbf{int}\}$  ist.

### Small step Semantik

$(\mathbf{let\ val\ } x = \mathit{ref\ } \{a = 0\} \mathbf{\ in\ } x := \{\}; \#a (!x) \mathbf{end}, [])$

steht für

$(\mathbf{let\ val\ } x = \mathit{ref\ } \{a = 0\} \mathbf{\ in\ } (\lambda u : \mathbf{unit}. \#a (!x)) (\mathit{:=\ } (x, \{\})) \mathbf{end}, [])$

$\rightarrow (\mathbf{let\ val\ } x = X \mathbf{\ in\ } (\lambda u : \mathbf{unit}. \#a (!x)) (\mathit{:=\ } (x, \{\})) \mathbf{end}, [X : \{a = 0\}])$   
mit Regel (REF)

$\rightarrow ((\lambda u : \mathbf{unit}. \#a (!X)) (\mathit{:=\ } (X, \{\})), [X : \{a = 0\}])$   
mit Regel (LET-EXEC)

$\rightarrow ((\lambda u : \mathbf{unit}. \#a (!X)) (), [X : \{\}])$   
mit Regel (ASSIGN)

$\rightarrow (\#a (!X), [X : \{\}])$   
mit Regel (BETA-V)

$\rightarrow (\#a \{\}, [X : \{\}])$   
mit Regel (DEREF)

und hier bleibt die Berechnung stecken, weil  $\{\}$  keine  $a$ -Komponente hat.

## Übungsblatt 12, Aufgabe 1 f

Das Programm ist *nicht* wohlgetypt, aber es terminiert trotzdem.

### Typüberprüfung

Bei der Deklaration kann  $x$  nur den Typ  $\{\}$  **ref** annehmen. Also ist die Zuweisung noch wohlgetypt (vgl. Aufgabe 1 a), aber die Projektion  $\#a (!x)$  nicht mehr.

### Small step Semantik

$(\text{let val } x = \text{ref } \{\} \text{ in } x := \{a = 0\}; \#a (!x) \text{ end}, [])$

steht für

$(\text{let val } x = \text{ref } \{\} \text{ in } (\lambda u:\text{unit}. \#a (!x)) (:= (x, \{a = 0\})) \text{ end}, [])$

$\rightarrow (\text{let val } x = X \text{ in } (\lambda u:\text{unit}. \#a (!x)) (:= (x, \{a = 0\})) \text{ end}, [X:\{\}])$   
mit Regel (REF)

$\rightarrow ((\lambda u:\text{unit}. \#a (!X)) (:= (X, \{a = 0\})), [X:\{\}])$   
mit Regel (LET-EXEC)

$\rightarrow ((\lambda u:\text{unit}. \#a (!X)) (), [X:\{a = 0\}])$   
mit Regel (ASSIGN)

$\rightarrow (\#a (!X), [X:\{a = 0\}])$   
mit Regel (BETA-V)

$\rightarrow (\#a \{a = 0\}, [X:\{a = 0\}])$   
mit Regel (DEREF)

$\rightarrow (0, [X:\{a = 0\}])$   
mit Regel (PROJ)

Sollte dieses Programm wohlgetypt sein? Nein, denn die Projektion  $\#a (!x)$  lässt sich nur deshalb auswerten, weil wir “zufällig” vorher einen passenden Wert in den Speicherplatz geschrieben haben.