Fortgeschrittene Konzepte der Programmierung UE 03

Steffen Anhuser, Jan-Erik Menzel, Melina Morch

Aufgabe 3.1

Restrukturieren Sie den Beweis zur Preservation aus der Vorlesung so, dass ber die Evaluationsregeln induziert wird, statt die Typregeln. Zu zeigen: Sei t:T und t -¿ t.
Dann gilt auch t:T Betrachte nur die Fall t = t1 t2 , da T-Tru, T-Fls etc. triviale Flle sind.
Inversion der Type Regeln liefert t1:T2 -¿ T und t2:T2 fr ein beliebiges T2. Wir erhalten 3 Flle und zeigen fr jede der Evaluationsregeln, dass t:T gilt.
Fall 1) E-App1: Dann gilt t = t1 t2 und t1 -¿ t1 Wegen t1:T2 -¿ T und t1 -¿ t1 gilt mit Induktionsvoraussetzung t1:T2 -¿ T Weiterhin gilt t2:T2, E-App fhrt zu t1t2:T , also t:T
Fall 2) E-App2: Der Fall ist hnlich zu Fall 1. Es gilt t = v1 t2 und t2 -¿ T2 Es gilt t2:T2, t2 -¿ t2 und die IV t2:T2 . Mit t1:T2 -¿ T folgt dann auch v1t2:T , also t:T
Fall 3) E-AppAbs: In diesem Fall sei t1 := x.t12 v2 fr ein v2 und t also [x2 -¿ b2]t12 fr ein b2. Wegen t1:T2 folgt x:T2 und b:T . Mit der IV t2:T2 erhalte dann durch Substitution t12:T , also t:T. In jedem Fall folgt also aus (t:T und t -¿ t) auch t:T.

Aufgabe 3.2

Zeigen Sie, dass jeder Subterm eines typsicheren Terms selbst typsicher ist.[1]
Ein typsicherer Term ist entweder ein Wert oder kann weiter reduziert werden (Progresseigenschaft). Wenn der Term einen Typ T hat und weiter reduzierbar ist, dann besitzt die Reduktion auch den Typen T (Preservation).

[1] Soll hier fr jeden zusammengesetzten Term ein Induktionsbeweis gefhrt werden? Sieht wahrscheinlich angenehmer aus als die Geschichte hier.

Die Menge der mglichen Typkombinationen ist eine Komposition von {T-True, T-False, T-If, T-Zero, T-Succ, T-Pred, T-IsZero}. Bei den Typregeln {T-True, T-False, T-Zero} ist der Term nicht mehr reduzierbar und damit automatisch typsicher. Nehmen wir an succ x ist typsicher mit dem Typen Nat. Jeder Typ Nat muss sich aus dem Term 0 und einer Kombination von succ, pred zusammensetzten . Diese sind aber immer auswertbar und geben mit den Typinferenzregeln den Typen mit. Damit kann es keinen sinnvollen Subterm x geben, welcher nicht selbst typsicher ist, sodass succ x oder pred x einen anderen Typen als x haben. Salopper ausgedrckt: Wenn ich annehme das succ x typsicher ist, muss x auch typsicher sein, da x eine Komposition von succ/pred seien muss oder der Wert 0. pred x funktioniert selber nur auf Nat und damit auch wieder auf einer Komposition von succ x, welches ja typsicher ist. iszero x setzt vorraus, dass x ein Nat ist und deswegen greift wieder derselbe Gedanke wie bei den anderen beiden Dingen. if x then y else z setzt einen Bool fr x vorraus. Dieser ist entweder iszero w (analoge Begrndung warum der typsicher ist) oder true oder false. true oder false sind beide typsicher. y und z knnen entweder ebenfalls einen Bool als Typen haben oder einen Nat. Fr Nat gilt die Begrndung von iszero, succ, pred, 0 und falls es ein Bool ist, muss er entweder iszero x oder false, true bzw. wieder eine if x then y else z sein. Fr letzteres gilt induktiv das gleiche wie vorher. Auch wenn die If-Schleifen unendlich oft miteinander verknpft werden, so muss am Ende trotzdem ein Nat oder Bool stehen.

Aufgabe 3.3

Zeigen Sie durch Aufstellen der Ableitungsbume, dass die folgenden Terme die jeweils angegeben Typen haben:

a)

$f : Bool \to Bool \vdash f(if\ false\ then\ true\ else\ false) : Bool$

Zuerst setzten wir $\Gamma := f : Bool \to Bool$ und $t_2 := if\ false\ then\ true\ else\ false.$

$\frac{\Gamma \vdash f{:}Bool\ \Gamma \vdash t_2{:}Bool}{\Gamma \vdash f\ t_2{:}\ Bool}$ (T-App)

$t_2 :\ \frac{\frac{}{false:Bool}(T-False)\ \frac{}{true:Bool}(T-True)\ \frac{}{false:Bool}(T-False)}{if\ false\ then\ true\ else\ false : Bool}$ (T-If)

b)

$f:\ Bool \to Bool \vdash \lambda x : Bool.f(if\ x\ then\ false\ else\ x) : Bool$

Wir gehen analog zur a) vor. Allerdings gibt es jetzt bei dem Term t_2 die Variable zu beachten und einzusetzen mit E-AppAbs. Also ist $\lambda x : Bool \vdash t_2 = if\ x\ then\ false\ else\ x$.

$\frac{\Gamma \vdash f : Bool\ \Gamma \vdash t_2 : Bool}{\Gamma \vdash f\ t_2 : Bool}$ (T-App)

$\lambda x : Bool \vdash t_2 : \frac{\frac{}{x:Bool}(T-Var)\ \frac{}{false:Bool}(T-false)\ \frac{}{x:Bool}(T-Var)}{if\ x\ then\ false\ else\ x : Bool}$ (T-If)