Semantik und Programmverifikation

Prof. Dr. Christoph Walther / Simon Siegler
Technische Universität Darmstadt — Wintersemester 2008/09

Hausaufgabe 6

Hausaufgabe 6.1 ($COND(t, \pi)$) **(4 Punkte)**

Zeigen Sie, dass für alle $t \in \mathcal{T}(\Sigma(P))$ gilt (vgl. Übung 3.1.4):

(a) $COND(t, \pi\rho) \approx COND(t, \pi) \wedge COND(t|_\pi, \rho)$ für alle $\pi \in Occ(t)$ und alle $\rho \in Occ(t|_\pi)$, vgl. Definition 1.2.2.

(b) $\mathsf{cbv\text{-}eval}_P(t) = \propto$ gdw. $COND(t, \pi) \Rightarrow_P^* TRUE$ und $\mathsf{cbv\text{-}eval}_P(t|_\pi) = \propto$ für ein $\pi \in Occ$.

Hausaufgabe 6.2 (Terminierung) **(3 Punkte)**

Seien F_{case_s}, F_f und F'_f Funktionsprozeduren mit $F_{case\_s} =$

function $case\_s(b : bool,\ x, y : s) : s \Leftarrow$ **if** b **then** x **else** y **fi**

für jede Sorte s. F'_f entsteht dabei aus F_f, indem jedes Funktionssymbol if_s in F_f durch $case\_s$ ersetzt wird. Formulieren Sie notwendige und hinreichende Bedingungen an F_f für

(a) F'_f terminiert *call-by-name*,

(b) F'_f terminiert *call-by-value*.

wenn alle anderen Funktionsprozeduren des Programms *call-by-value* terminieren (vgl. Übung 3.1.7).

Hausaufgabe 6.3 (Primitiv-rekursive Funktionen) **(2 Punkte)**

Wir nennen eine Funktionsprozedur F *primitiv-rekursiv* gdw. F nicht rekursiv definiert oder aber von der Form

function $f(x : nat, y_1 : s_1, \ldots, y_k : s_k) : s \Leftarrow$
  if $x = 0$
    then $g(y_1, \ldots, y_k)$
    else $h(f(x-1, y_1, \ldots, y_k), x-1, y_1, \ldots, y_k)$
  fi

ist, wobei $g \in \Sigma(BM)$ oder **function** $g(y_1 : s_1, \ldots, y_k : s_k) : s \Leftarrow \ldots$ primitiv-rekursiv ist und ebenso $h \in \Sigma(BM)$ oder **function** $h(z : s, x : nat, y_1 : s_1, \ldots, y_k : s_k) : s \Leftarrow \ldots$ primitiv-rekursiv ist. Ein funktionales Programm P heißt dementsprechend *primitiv-rekursiv* gdw. jede Funktionsprozedur in P primitiv-rekursiv ist. Beweisen Sie, dass jedes primitiv-rekursive Programm stark terminiert.