Script generated by TTT

Title: Meixner: ZUE_THEO (26.06.2014)

Date: Thu Jun 26 15:06:11 CEST 2014

Duration: 45:14 min

Pages: 22

SS 2014

Zentralübung zur Vorlesung Theoretische Informatik

Dr. Werner Meixner

Fakultät für Informatik
TU München

http://www14.in.tum.de/lehre/2014SS/theo/uebung/

26. Juni 2014

ZÜ VII

Übersicht:

1. Übungsbetrieb Fragen, Probleme? Klausur!
2. Thema Erweiterte PR Regeln
3. Vorbereitung Blatt 10

2. Thema Erweiterte PR Regeln

Die Konstruktion von primitiv-rekursiven Funktionen stützt sich wesentlich auf die Erzeugungsregeln für funktionale Ausdrücke.

Die Regeln dafür kann man weit oder eng fassen in Abhängigkeit davon, welchen Schwerpunkt man auf die Beweise legt.

Die sogenannten erweiterten PR Regeln lassen in gewissem Sinne die Bildung aller funktionalen Ausdrücke zu.

Siehe Vorbereitungsaufgaben

3. Vorbereitung Blatt 10

3.1 VA 1

Vergleichen Sie die beiden folgenden WHILE-Programme:

1. WHILE $b_0 \wedge b_1$ DO P END
2. WHILE b_0 DO P END; WHILE $b_0 \wedge b_1$ DO P END.

Zeigen diese Programme das gleiche Verhalten? Begründung!

3. Vorbereitung Blatt 10

3.1 VA 1

Vergleichen Sie die beiden folgenden WHILE-Programme:

1. WHILE $b_0 \wedge b_1$ DO P END
2. WHILE b_0 DO P END; WHILE $b_0 \wedge b_1$ DO P END.

Zeigen diese Programme das gleiche Verhalten? Begründung!

Lösung

Die beiden Programme zeigen nicht das gleiche Verhalten!

Seien $b_0 =$ true und $b_1 =$ false, als Konstante 1 bzw. 0.

Dann terminiert das Programm 1, ohne die Variablen zu verändern.

Das Programm 2 terminiert nicht, wenn man voraussetzt,
dass P die Variable b_1 nicht verändert, d.h. auf true setzt.

Problem

Was ist das „Verhalten" eines Programms?
Welche Semantik benutzt man hier?

3. Vorbereitung Blatt 10

3.1 VA 1

Vergleichen Sie die beiden folgenden WHILE-Programme:

1. WHILE $b_0 \wedge b_1$ DO P END
2. WHILE b_0 DO P END; WHILE $b_0 \wedge b_1$ DO P END.

Zeigen diese Programme das gleiche Verhalten? Begründung!

Lösung

Die beiden Programme zeigen nicht das gleiche Verhalten!

Seien $b_0 =$ `true` und $b_1 =$ `false`, als Konstante 1 bzw. 0.

Dann terminiert das Programm 1, ohne die Variablen zu verändern.

Das Programm 2 terminiert nicht, wenn man voraussetzt,
dass P die Variable b_1 nicht verändert, d.h. auf `true` setzt.

Problem

Was ist das „Verhalten" eines Programms?
Welche Semantik benutzt man hier?

3. Vorbereitung Blatt 10

3.1 VA 1

Vergleichen Sie die beiden folgenden `WHILE`-Programme:

1. `WHILE` $b_0 \wedge b_1$ `DO` P `END`
2. `WHILE` b_0 `DO` P `END`; `WHILE` $b_0 \wedge b_1$ `DO` P `END`.

Zeigen diese Programme das gleiche Verhalten? Begründung!

Lösung

Die beiden Programme zeigen nicht das gleiche Verhalten!

Seien $b_0 =$ `true` und $b_1 =$ `false`, als Konstante 1 bzw. 0.

Dann terminiert das Programm 1, ohne die Variablen zu verändern.

Das Programm 2 terminiert nicht, wenn man voraussetzt,
dass P die Variable b_1 nicht verändert, d.h. auf `true` setzt.

Problem

Was ist das „Verhalten" eines Programms?
Welche Semantik benutzt man hier?

3.2 VA 2

Zeigen Sie durch Rückführung auf die Definition, dass die folgenden Funktionen primitiv-rekursiv sind:

$$iszero(x) = \begin{cases} 1 & \text{falls } x = 0 \\ 0 & \text{sonst} \end{cases} \quad , \quad eq(x,y) = \begin{cases} 1 & \text{falls } x = y \\ 0 & \text{sonst} \end{cases} \quad .$$

Lösung

Wir schreiben die primitiv-rekursiven Basis-Projektionsfunktionen $proj_{k,i}(x_1, x_2, \ldots, x_k)$ als $\pi_i^k(x_1, x_2, \ldots, x_k)$.

s sei die Basis-Nachfolgerfunktion.

Wir definieren zunächst die primitiv-rekursiven arithmetischen Funktionen $add(x,y)$, $mult(x,y)$, $\dot{-}(x,y)$ bzw. in üblicher Infixschreibweise $x+y$, $x \cdot y$, $x \dot{-} y$

und Semantik

$$\dot{-}(x,y) = \max\{x-y, 0\}.$$

Addition

Lesbare Kurzschrift zuerst:

$$\begin{aligned} 0 + y &= y\,, \\ (x+1) + y &= s(x+y)\,. \end{aligned}$$

oder

$$\begin{aligned} add(0,y) &= y\,, \\ add(x+1,y) &= s(add(x,y)) \end{aligned}$$

Syntaktisches Format:

$$h(r,x,y) = s(\pi_1^3(r,x,y))\,,$$

$$\begin{aligned} add(0,y) &= \pi_1^1(y)\,, \\ add(x+1,y) &= h(add(x,y),x,y)\,. \end{aligned}$$

Multiplikation

Lesbare Kurzschrift zuerst:

$$\begin{aligned} 0 \cdot y) &= 0\,, \\ (x+1) \cdot y &= (x \cdot y) + y\,. \end{aligned}$$

oder

$$\begin{aligned} mult(0,y) &= 0\,, \\ mult(x+1,y) &= add(mult(x,y),y) \end{aligned}$$

Syntaktisches Format:

$$\begin{aligned} k(y) &= 0\,, \\ h(r,x,y) &= add(\pi_1^3(r,x,y), \pi_3^3(r,x,y))\,, \end{aligned}$$

$$\begin{aligned} mult(0,y) &= k(y)\,, \\ mult(x+1,y) &= h(mult(x,y),x,y)\,. \end{aligned}$$

Modifizierte Subtraktion

Lesbare Kurzschrift zuerst:

$$\begin{aligned} pred(0) &= 0\,, \\ pred(x+1) &= x\,, \end{aligned}$$

$$\begin{aligned} x \dot{-} 0 &= x\,, \\ x \dot{-} (y+1) &= pred(x \dot{-} y)\,. \end{aligned}$$

Syntaktisches Format:

$$
\begin{aligned}
h_1(r,x) &= \pi_2^2(r,x)\,,\\
pred(0) &= 0\,,\\
pred(x+1) &= h_1(pred(x),x)\,,\\
h_2(r,x,y) &= pred(\pi_1^3(r,x,y))\,,\\
\dot{-}^R(0,x) &= \pi_1^1(x)\,,\\
\dot{-}^R(y+1,x) &= h_2(\dot{-}^R(y,x),y,x)\,,\\
\dot{-}(x,y) &= \dot{-}^R(y,x)\,.
\end{aligned}
$$

Syntaktisches Format:

$$
\begin{aligned}
h_1(r,x) &= \pi_2^2(r,x)\,,\\
pred(0) &= 0\,,\\
pred(x+1) &= h_1(pred(x),x)\,,\\
h_2(r,x,y) &= pred(\pi_1^3(r,x,y))\,,\\
\dot{-}^R(0,x) &= \pi_1^1(x)\,,\\
\dot{-}^R(y+1,x) &= h_2(\dot{-}^R(y,x),y,x)\,,\\
\dot{-}(x,y) &= \dot{-}^R(y,x)\,.
\end{aligned}
$$

Lösung in lesbarer Form:

$$
\begin{aligned}
iszero(0) &= 1\,,\\
iszero(x+1) &= 0\,,\\
eq(x,y) &= iszero((x \dot{-} y) + (y \dot{-} x))\,.
\end{aligned}
$$

Mit welchen Regeln kann man stets eine lesbare Form erreichen?

Dieser Frage werden wir in VA 4 nachgehen.

3.3 VA 3

Sei $f(x,y)$ primitiv rekursiv.

Zeigen Sie mit Hilfe der Projektionsfunktionen π_i^k zusammen mit der (nicht erweiterten) Komposition, dass die Funktion g mit

$$g(x,y) = f(y,x)$$

für alle $x, y \in \mathbb{N}$ ebenfalls primitiv rekursiv ist.

Lösung

$$g(x,y) = f(\pi_2^2(x,y), \pi_1^2(x,y))\,.$$

3.4 VA 4

Wir bezeichnen f als eine *erweiterte Komposition* der Funktionen $g_1, \ldots, g_k$, falls

$$f(x_1, \ldots, x_n) = t\,,$$

so dass t ein Ausdruck ist, der nur aus den Funktionen $g_1, \ldots, g_k$ und den Variablen $x_1, \ldots, x_n$ besteht.

Beispiel:

$$f(x, y) = g_1(x, g_2(y, g_3(x)))\,.$$

Sei t_0 ein funktionaler Ausdruck, der nur primitiv-rekursive Funktionen und Variable x_i enthält.
t enthalte als Teilausdrücke nur $f(m, \overline{x})$ mit einem Variablenvektor $\overline{x}$, primitiv-rekursive Funktionen, m und Variable x_i.

Dann heißen die Gleichungen

$$f(0, \overline{x}) = t_0\,, \quad f(m + 1, \overline{x}) = t$$

das *erweiterte Schema* der primitiven Rekursion.

Beweis

Das folgende LOOP - Programm stützt sich auf das LOOP-Programm IF $x = 0$ THEN P END, das in der Vorlesung simuliert wurde.

Seien $x_1, x_2, \notin \{x_i, x_j\}$.

$x_1 := x_i$; LOOP x_j DO $x_1 := x_1 - 1$;   // $x_1 = x_i \dot{-} x_j$
IF $x_1 = 0$ THEN P_1 ; $x_2 := 1$ END ;
IF $x_2 = 0$ THEN P_2 END ;