08.11.05

Theorie der Programmierung I

(Mitschrift von Lars Friedrich email@lars-friedrich-home.de)

Big step: $e \Downarrow r$ mit $r \in Val \cup Exn$

(COND-TRUE) $\dfrac{e_0 \Downarrow \text{true}\ \ e_1 \Downarrow v}{\text{if } e_0 \text{ then } e_1 \text{ else } e_2 \Downarrow v}$

(COND-FALSE) $\dfrac{e_0 \Downarrow \text{false}\ \ e_2 \Downarrow v}{\text{if } e_0 \text{ then } e_1 \text{ else } e_2 \Downarrow v}$

(UNFOLD) $\dfrac{e[\text{rec id } e/id] \Downarrow v}{\text{rec id } e \Downarrow v}$

Meta-Regel für exception:

Zu jeder Regel

(R) $\dfrac{e_1 \Downarrow v_1 \ldots e_n \Downarrow v_n}{e \Downarrow n}$ fügt man n exception-Regeln hinzu:

(R-EXN-1) $\dfrac{e_1 \Downarrow exn}{e \Downarrow exn}$ … **(R-EXN-N)** $\dfrac{e_1 \Downarrow v1 \ldots e_{n-1} \Downarrow v_{n-1}\ e_n \Downarrow exn}{e \Downarrow exn}$

z.B.:

(COND-TRUE-EXN-1) $\dfrac{e_0 \Downarrow exn}{\text{if } e_0 \text{ then } e_1 \text{ else } e_2 \Downarrow exn}$

(COND-TRUE-EXN-2) $\dfrac{e_0 \Downarrow \text{true}\quad e_1 \Downarrow exn}{\text{if } e_0 \text{ then } e_1 \text{ else } e_2 \Downarrow exn}$

Abgeleitete big step Regel für let-Ausdrücke:

(LET) $\dfrac{e_1 \Downarrow v_1 \qquad e_2[v_1/id] \Downarrow v_2}{\text{let id} = e_1 \text{ in } e_2 \Downarrow v_2}$

Herleitung dieser Regel: let id = e_1 in e_2 steht für $(\lambda id.\ e_2)\ e_1$

Also zu zeigen: Aus den Prämissen (LET) erhält man $(\lambda id.\ e_2)\ e_1 \Downarrow v_2$.

(1) $(\lambda id.\ e_2)\ e_1 \Downarrow v_2$

(APP)

↳ (2) $\lambda id.\ e_2 \Downarrow \lambda id.\ e_2$ wegen (VAL)

↳ (3) $e_1 \Downarrow v_1$ wegen 1. Prämisse

↳ (4) $(\lambda id.\ e_2)\ v_1 \Downarrow v_2$

(BETA-V)

↳ (5) $e_2\ [v_1/id] \Downarrow v_2$ wegen 2. Prämisse

Beispiel-Programm:

(1) let fact = rec fact λx. if = x 0 then 1 else ∗ x (fact (- x 1)) in fact 1 ⇓ 1

(LET)

↳ (2) rec fact λx. if = x 0 then 1 else ∗ x (fact (- x 1)) ⇓ λx. if = x 0 then …

(UNFOLD)

↳ (3) λx. if = x 0 then 1 else ∗ x (fact (- x 1))⇓ λx. if = x 0 then … (VAL)

↳ (4) (λx. if = x 0 then 1 else ∗ x (rec … (- x 1))) 1 ⇓ 1

(BETA-V)

↳ (5) if = 1 0 then 1 else ∗ 1 (rec … (- 1 1)) ⇓ 1

(COND-FALSE)

↳ (6) = 1 0 ⇓ false wegen (OP)

↳ (7) ∗ 1 (rec … (- 1 1)) ⇓ 1

(APP)

↳ (8) ∗ 1 ⇓ ∗ 1 (VAL)

↳ (9) (rec …) (- 1 1) ⇓ 1

(APP)

↳ (10) rec … ⇓ λx. if = x 0 …

(UNFOLD)

↳ (11) λx. if = x 0 … ⇓ λx. if = x 0 …

↳ (12) - 1 1 ⇓ 0

↳ (13) (λx. if = x 0 …) 0 ⇓ 1

(BETA-V)

↳ (14) if = 0 0 then 1 else … ⇓ 1

(COND-TRUE)

↳ (15) = 0 0 ⇓ true

↳ (16) 1 ⇓ 1

↳ (17) ∗ 1 1 ⇓ 1

Satz 4: Für alle e ∈ Exp und r ∈ Val ∪ EXN gilt: $e \xrightarrow{*} r \Leftrightarrow e \Downarrow r$

Schreibweise: $e \not\Downarrow$ bedeutet: Es existiert kein r mit e ⇓ r

Folgerung aus Satz 4:

(a) $e \not\Downarrow$ ⇔ die Berechnung für e divigiert oder bleibt stecken (d.h.) in der big step Schreibweise wird zwischen Divergenz und stecken bleiben nicht unterschieden)

(b) Für jeden Ausdruck e existiert höchstens ein r ∈ Val ∪ Exn mit e ⇓ r

(c) Wenn e ⇓ r, dann free(r) ⊆ free(e)

Beweis von Satz 4:

Hier: nur r ∈ Val, d.h. wir ignorieren die exception-Regeln

zu zeigen: $e \xrightarrow{*} r \Leftrightarrow e \Downarrow v$

„⇐“: Idee: Man orientiert sich an den big step Regeln. Nach Induktionsannahme gibt es für die big steps in der Prämisse entsprechende Folgen von small steps. Diese small step Folgen setzt man zusammen, um eine small step Folge für die Konklusion zu erhalten.

Beweistechnik: Induktion über die Länge der Herleitung des big steps e ⇓ v und Fallunterscheidung nach der zuletzt angewandten big step Regel.

Voraussetzung: e ⇓ v

zu zeigen: $e \xrightarrow{*} v$

Induktionsanfang:

(VAL) wenn $e \Downarrow v$ wegen (VAL), dann ist e = v (syntaktisch). Es gilt: $v \xrightarrow{*} v$.

(OP) $op\ n_1\ n_2 \Downarrow v$ wegen (OP), dann ist $v = op^I(n_1\ n_2)$, also gilt $op\ n_1\ n_2 \xrightarrow{*} v$ mit small step Regel (OP).

Induktionsschritt:

(BETA-V) wenn $(\lambda id.\ e)\ v \Downarrow v'$ mit (BETA-V) aus $e[v/id] \Downarrow v'$ folgt, dann gilt nach Induktionsannahme $e[v/id] \xrightarrow{*} v'$. Also gilt mit small step Regel (BETA-V): $(\lambda id.\ e)v \rightarrow e\ [v/id] \xrightarrow{*} v'\ \}\ (\lambda id.\ e)\ v \xrightarrow{*} v'$

(APP) wenn $e_1e_2 \Downarrow v$ mit (APP) aus $e_1 \Downarrow v_1$, $e_2 \Downarrow v_2$ und $v_1v_2 \Downarrow v$ folgt, dann gilt nach Induktionsannahme $e_1 \xrightarrow{*} v_1$, $e_2 \xrightarrow{*} v_2$ und $v_1v_2 \xrightarrow{*} v$. Also gilt:

$e_1e_2 \xrightarrow{*} v_1e_2$ | mit (APP-LEFT) aus $e_1 \xrightarrow{*} v_1$ $\xrightarrow{*} v_1v_2$ | mit (APP-RIGHT) aus $e_2 \xrightarrow{*} v_2$ $\xrightarrow{*} v$ | wie oben

übrige Regeln analog!

„⇒“:

Voraussetzung: $e \xrightarrow{*} v$

zu zeigen: $e \Downarrow v$

Idee: Die Folge $e \xrightarrow{*} v$ muss man „geschickt“ in Teilfolgen zerlegen. Für die Teilfolgen existieren nach Induktionsannahme big steps. Diese big steps lassen sich dann wieder zusammensetzen.