Denotationale Programmsemantik

4.27 Definition

Sei A eine (S, Σ)-Algebra, V Variablenmenge, $z, z' \in \mathcal{Z}(A, V)$ (Zustände über A, V) und α ein Programm über (S, Σ) und V.

Induktiv über den Aufbau eines Programms wird definiert, was es heißt, dass der Zustand z durch Abarbeitung von α in den Zustand z' überführt wird, notiert als $z[[\alpha]]_A z'$, d. h. α bezeichnet eine Relation $[[\alpha]]_A$ auf $\mathcal{Z}(A, V)$.

- $z[[\varepsilon]]_A z' \text{ gdw } z = z'$.
- $z[[X := t]]_A z' \text{ gdw } z' = z(X/val_{A,z}(t))$
- $z[[$**if** B **then** β **else** γ **end**; $]]_A z' \text{ gdw}$
  $(A \models_z B$ und $z[[\beta]]_A z')$ oder $(A \not\models_z B$ und $z[[\gamma]]_A z')$.
- $z[[$**while** B **do** β **end**; $]]_A z' \text{ gdw}$
  es gibt eine Zahl $n \in \mathbb{N}$ und Zustände $z_0, \ldots, z_n$ mit
  - $z = z_0$,
  - $A \models_{z_i} B$ und $z_i[[\beta]]_A z_{i+1}$ für $0 \leq i < n$
  - $A \not\models_{z_n} B$
  - $z_n = z'$
- $z[[\alpha\beta]]_A z' \text{ gdw}$ es gibt einen Zustand z_1 mit $z[[\alpha]]_A z_1$ und $z_1[[\beta]]_A z'$.

Beachte: Die zweistellige Relation $[[\alpha]]_A$ ist rechtseindeutig aber nicht immer rechtsvollständig, d. h. aus $z[[\alpha]]_A z'$ und $z[[\alpha]]_A z''$ folgt $z' = z''$, aber nicht zu jedem α und z muss es ein z' geben mit $z[[\alpha]]_A z'$.

Beispiele

Bei einer While-Schleife beschreibt $[[-]]_A$ den Ablauf.

$z = z_0$	$[[\beta]]_A$	z_1	$[[\beta]]_A$	$\cdots$	z_{n-1}	$[[\beta]]_A$	$z_n = z'$
B gilt	$\cdots$	gilt		$\cdots$	gilt		B gilt nicht mehr

4.28 Beispiel Im Beispiel 4.26: $z[[\alpha]]_{Nat}z'$ und sei $z(X) = 2$:

$\alpha :: z(X) = 2, z(Y) = 0, z(Z) = 0$

- $Nat \models_z \neg Y = X$
  $z_1(X) = 2, z_1(Y) = 1, z_1(Z) = 1$
- $Nat \models_{z_1} \neg Y = X$
  $z_2(X) = 2, z_2(Y) = 2, z_2(Z) = 4$
- $Nat \not\models_{z_2} \neg Y = X$
  $z' = z_2$

Allgemein gilt:

$z'(Z) = z(X)^2$
$z'(X) = z(X)$
$z'(Y) = z(X)$

α „berechnet“ mit Eingabe X in Z die Funktion $f(x) = x^2$.

Analog β berechnet in Z die Funktion $f(x, y) = x + y$.

Analog γ berechnet in Z die Funktion $f(x, y) = x * y$.

D. h. $+_{nat}, *_{nat}$ sind über N „berechenbar“ durch While-Programme.

Interpretersemantik

4.29 Definition Sei A eine (S, Σ)-Algebra und V eine Variablenmenge. Die Interpreterfunktion I_A ist eine zweistellige totale Funktion, die einem Programm α und Zustand z, das Programm α' und den Zustand z' zuordnet, $(d.h. I_A : (Prog, \mathcal{Z}) \to (Prog, \mathcal{Z}))$, die sich nach Abarbeitung der ersten Anweisung von α im Zustand z als Restprogramm und neuer Zustand ergeben. Sie wird induktiv wie folgt definiert:

- $I_A(\varepsilon, z) = (\varepsilon, z)$.
- $I_A(X := t;\ \beta, z) = (\beta, z(X/val_{A,z}(t)))$.
- $I_A(\underline{\textbf{if}}\ B\ \underline{\textbf{then}}\ \gamma\ \underline{\textbf{else}}\ \delta\ \underline{\textbf{end}};\ \beta, z) = \begin{cases} (\gamma\beta, z) & \text{falls } A \models_z B \\ (\delta\beta, z) & \text{sonst} \end{cases}$
- $I_A(\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \gamma\ \underline{\textbf{end}};\ \beta, z) =$
  $\begin{cases} (\gamma \underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \gamma\ \underline{\textbf{end}};\ \beta, z) & \text{falls } A \models_z B \\ (\beta, z) & \text{sonst} \end{cases}$

I_A ist wohldefiniert und total auf der Menge $(Prog, \mathcal{Z})$.

Beachte in $I_A(\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \gamma\ \underline{\textbf{end}};\ \beta, z)$ ist „Restprogramm“ strukturell komplexer als Ausgangsprogramm (im Fall $A \models_z B$).

Beispiel (Fort.)

4.30 Beispiel Im Beispiel 4.26: While-Programm $\alpha = S_1 S_2 S_3$ über Signatur von Nat.

$z(X) = 2, z(Y) = 0, z(Z) = 3$:
Iteration von I_A.

$$
\begin{aligned}
I_A^9(\alpha, z) &= I_A^9(S_1 S_2 S_3, z) = I_A^7(S_3, z(Y/0, Z/0)) \\
&= I_A^6(Z := succ(Z + (Y + Y)); \\
&\qquad\qquad Y := succ(Y);\ S_3, z(Y/0, Z/0)) \\
&= I_A^5(Y := succ(Y);\ S_3, z(Y/0, Z/1)) \\
&= I_A^4(S_3, z(Y/1, Z/1)) \\
&= I_A^3(Z := succ(Z + (Y + Y)); \\
&\qquad\qquad Y := succ(Y);\ S_3, z(Y/1, Z/1)) \\
&\ldots \\
&= I_A(S_3, z(Y/2, Z/4)) \\
&= (\varepsilon, z(Y/2, Z/4))
\end{aligned}
$$

D. h. $I_A^9(\alpha, z) = (\varepsilon, z')$ mit z' wie gehabt.
Offenbar gilt $z[[\alpha]]_A z'$ für dieses Beispiel.

Äquivalenz der Semantikbegriffe

4.31 Lemma Sei A eine (S, Σ)-Algebra, V eine Variablenmenge, $z, z' \in \mathcal{Z}(A, V)$ Zustände und α ein Programm über (S, Σ) und V. Dann gilt

$$z[[\alpha]]_A z' \text{ gdw } \exists n \in \mathbb{N}^+ : I_A^n(\alpha, z) = (\varepsilon, z')$$

Beweis: Induktion über Aufbau von α.

- α Zuweisung $X := t;$, t Term mit Typ(X) = Typ(t)
  $z[[\alpha]]_A z' \quad \text{gdw} \quad z' = z(X/val_{A,z}(t))$
  $\text{gdw} \quad I_A(\alpha, z) = (\varepsilon, z')$ (d. h. $n = 1$ gewählt)
- α Test **if** B **then** β **else** γ **end**; analog.
- α Schleife **while** B **do** β **end**; und die Behauptung gelte für β.
  Es gelte $z[[\alpha]]z'$, dann
  $z[[$**while** B **do** β **end**; $]]_A z'$ gdw
  es gibt eine Zahl $m \in \mathbb{N}$ und Zustände $z_0, \ldots, z_m$ mit
  - $z = z_0$,
  - $A \models_{z_i} B$ und $z_i[[\beta]]_A z_{i+1}$ für $0 \leq i < m$
  - $A \not\models_{z_m} B$
  - $z_m = z'$

  Wähle minimale Zahlen $n_i (i = 0, ..., m-1)$ mit $I_A^{n_i}(\beta, z_i) = (\varepsilon, z_{i+1})$. Mit $n := n_0 + ... + n_{m-1}$ folgt die Behauptung. Umkehrung?.

Äquivalenz der Semantikbegriffe (Forts.)

- $\alpha = \beta\gamma$ o.B.d.A. $\beta, \gamma \neq \varepsilon$ Angenommen $z[[\alpha]]_A z'$, dann gibt es einen Zustand z_1 mit $z[[\beta]]z_1$ und $z_1[[\gamma]]z'$. Nach Induktionsvoraussetzung gibt es Zahlen n_1, n_2, so dass
  $I_A^{n_1}(\beta, z) = (\varepsilon, z_1)$ und $I_A^{n_2}(\gamma, z_1) = (\varepsilon, z')$.

Wähle n_1, n_2 minimal mit dieser Eigenschaft. Dann gilt mit $n = n_1 + n_2$:

$$\begin{aligned} I_A^n(\alpha, z) &= I_A^{n_1+n_2}(\beta\gamma, z) = I_A^{n_2}(I_A^{n_1}(\beta\gamma, z)) \\ &= I_A^{n_2}(\gamma, z_1) = (\varepsilon, z') \end{aligned}$$

Wo benötigt man die Minimalität von n_1?

Sei umgekehrt $z[[\alpha]]_A z'$ nicht gültig. Entweder gilt dann $z[[\alpha]]_A z''$ für einen anderen Zustand, insbesondere auch $I_A^n(\alpha, z) = (\varepsilon, z'')$ für ein n wie eben gezeigt, und also für kein n' $I_A^{n'}(\alpha, z) = (\varepsilon, z')$; oder es gibt keinen Zustand z', so dass $z[[\alpha]]_A z'$ gilt. Dann gibt es entweder z'' mit $z[[\beta]]z''$ aber keinen Zustand z' mit $z[[\gamma]]z'$ oder keinen Zustand z'' mit $z[[\beta]]z''$.

Die induktive Voraussetzung liefert im ersten Fall: $I_A^n(\beta, z) = (\varepsilon, z'')$ für ein n und für kein n' kann I_A γ in n' auf den Zustand z'' zum leeren Programm abarbeiten. Dann kann aber auch kein n'' existieren, so dass I_A $\beta\gamma$ auf den Zustand z zum leeren Programm abgearbeitet wird.

Analog im zweiten Fall.

While-Berechenbare Funktionen

4.32 Definition

Sei A eine (S, Σ)-Algebra. Eine Funktion $f : s_A^1 \times \cdots \times s_A^n \longrightarrow s_A$ heißt (while-) **programmierbar** (while-berechenbar), falls es eine Variablenmenge V über (S, Σ), die die paarweise verschiedenen Variablen X_i vom Typ s^i (Eingabevariablen) und Y vom Typ s (Ausgabevariable) enthält, und ein Programm α über (S, Σ) und V gibt, so dass für alle Zustände $z \in \mathcal{Z}(A, V)$ gilt:

$f(z(X_1), \ldots, z(X_n)) \downarrow$
$\rightsquigarrow$ es gibt ein $z' \in \mathcal{Z}(A, V)$ mit
$z[[\alpha]]_A z'$ und $z'(Y) = f(z(X_1), \ldots, z(X_n))$.

$f(z(X_1), \ldots, z(X_n)) \uparrow$
$\rightsquigarrow$ es gibt kein $z' \in \mathcal{Z}(A, V)$ mit $z[[\alpha]]_A z'$.

4.33 Beispiel Beispiel 4.26 (Fort.)

Programm α berechnet die Funktion $f(x) = x^2$ in Nat mit
Eingabevariable X Ausgabevariable Z

Programm β berechnet die Funktion $f(x, y) = x +_{Nat} y$ in N mit
Eingabevariable X, Y Ausgabevariable Z

Programm γ berechnet die Funktion $f(x, y) = x *_{Nat} y$ in N mit
Eingabevariable X, Y Ausgabevariable Z