5. Übung zur
Konstruktion fehlerfreier Software

Frank Pilhofer

Montag 13-14 Uhr

1. Auswertung einer while-Anweisung

Sei $d^0 = \left[(i,Z,0),(s,R,0),(n,Z,2),(x(1),Z,1),(x(2),Z,2)\right]$.

$$\begin{pmatrix} \text{while} & i<n & \text{do} \\ & i:=i+1 & \\ & s:=s+x(i) & \\ \text{endwhile} & & \end{pmatrix} \cdot \begin{bmatrix} (i,Z,0),(s,R,0) \\ (n,Z,2),(x(1),Z,1),(x(2),Z,2) \end{bmatrix}$$

$$= \left(\begin{pmatrix} i:=i+1 \\ s:=s+x(i) \end{pmatrix}, \begin{pmatrix} \text{while} & i<n & \text{do} \\ & i:=i+1 & \\ & s:=s+x(i) & \\ \text{endwhile} & & \end{pmatrix} \right) \cdot \begin{bmatrix} (i,Z,0),(s,R,0) \\ (n,Z,2),(x(1),Z,1),(x(2),Z,2) \end{bmatrix}$$

$$= \begin{pmatrix} \text{while} & i<n & \text{do} \\ & i:=i+1 & \\ & s:=s+x(i) & \\ \text{endwhile} & & \end{pmatrix} \cdot \begin{bmatrix} (i,Z,1),(s,R,1) \\ (n,Z,2),(x(1),Z,1),(x(2),Z,2) \end{bmatrix}$$

$$= \left(\begin{pmatrix} i:=i+1 \\ s:=s+x(i) \end{pmatrix}, \begin{pmatrix} \text{while} & i<n & \text{do} \\ & i:=i+1 & \\ & s:=s+x(i) & \\ \text{endwhile} & & \end{pmatrix} \right) \cdot \begin{bmatrix} (i,Z,1),(s,R,1) \\ (n,Z,2),(x(1),Z,1),(x(2),Z,2) \end{bmatrix}$$

$$= \begin{pmatrix} \text{while} & i<n & \text{do} \\ & i:=i+1 & \\ & s:=s+x(i) & \\ \text{endwhile} & & \end{pmatrix} \cdot \begin{bmatrix} (i,Z,2),(s,R,3) \\ (n,Z,2),(x(1),Z,1),(x(2),Z,2) \end{bmatrix}$$

$$= \begin{bmatrix} (i,Z,2),(s,R,3) \\ (n,Z,2),(x(1),Z,1),(x(2),Z,2) \end{bmatrix}$$

1. Definitionsbereich der while-Anweisung

Damit die while-Anweisung definiert ist, muß

- die Bedingung in jeder Iteration definiert (und vom Typ Boolean) sein
- das Programmstück in jeder Iteration definiert sein und
- nach endlich vielen Iterationen muß die Bedingung falsch werden.

Also:

$$\mathrm{Def}_{\text{w}\,B\,\text{do}\,S\,\text{end}} = \left\{ d \in D \,\middle|\, \exists n: \left(B.\left(S^n.d\right) \wedge \overline{\left(\bigvee_{j=0}^{n-1} B.\left(S^j.d\right) \right)} \wedge \left(\bigwedge_{j=0}^{n} S^j.d \in \mathrm{Def}_B \right) \wedge \left(\bigwedge_{j=0}^{n-1} S^j.d \in \mathrm{Def}_S \right) \right) \right\}$$

2. Deklarationen

Sei $d = [(x, R, 9.3), (y, R, 2)]$

2.0. declare(z,R,x+y)

$$(\text{declare}(z, R, x+y)).[(x, R, 9.3), (y, R, 2)]$$
$$= \quad (\text{declare}(z, R, (x+y).[(x, R, 9.3), (y, R, 2)])).[(x, R, 9.3), (y, R, 2)]$$
$$= \quad (\text{declare}(z, R, 11.3)).[(x, R, 9.3), (y, R, 2)]$$
$$= \quad [(z, R, 11.3), (x, R, 9.3), (y, R, 2)]$$

2.1. declare(z,R,z+y)

Diese Deklaration ist nicht definiert, da z in d nicht definiert ist.

3. Definitionsbereich von declare

Die Deklaration $x := A$ ist definiert, wenn

- der Ausdruck definiert ist
- das Ergebnis von A in d mit dem Wertebereich von x kompatibel ist.

Also: $\text{Def}_{\text{declare}(x,W,A)} = \{d \in D \mid d \in \text{Def}_A \wedge A.d \in W\}$.

4. Declare und Release

Sei $d = [(x, R, 9.3), (y, R, 2)]$

4.0. declare(z,R,x+y); release z

$$(\text{declare}(z, R, x+y), \text{release } z).[(x, R, 9.3), (y, R, 2)]$$
$$= \quad (\text{release } z).[(z, R, 11.3), (x, R, 9.3), (y, R, 2)] \quad = \quad [(x, R, 9.3), (y, R, 2)]$$

4.1. declare(z,R,x+2y); release y

$$(\text{declare}(z, R, x+2y), \text{release } y).[(x, R, 9.3), (y, R, 2)]$$
$$= \quad (\text{release } y).[(z, R, 13.3), (x, R, 9.3), (y, R, 2)] \quad = \quad [(z, R, 13.3), (x, R, 9.3)]$$

4.2. declare(z,R,x); release x

$$(\text{declare}(z, R, x), \text{release } x).[(x, R, 9.3), (y, R, 2)]$$
$$= \quad (\text{release } x).[(z, R, 9.3), (x, R, 9.3), (y, R, 2)] \quad = \quad [(z, R, 9.3), (y, R, 2)]$$

5. Definitionsbereich von Release

Damit Release definiert ist, muß in der Datenumgebung eine Variable dieses Namens existieren.

Also: $\text{Def}_{\text{release } x} = \{d^j \in D \mid \exists i: N_i^j = x\}$.