Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Clemens Ballarin

Vorlesung Semantik
Sommersemester 2005
Übungsblatt 5
Besprechung am 20. Mai 2005

Aufgabe 1 (H) (*Definite Assignment*)

Wird in einer Programmiersprache *Definite Assignment* verlangt, bedeutet dies die Forderung, dass alle Variablen vor ihrer ersten Benutzung auch initialisiert werden. Das Statement $x := 1$ ist zum Beispiel also auf jeden Fall zulässig, $x := y$ dagegen darf nur zugelassen werden, wenn feststeht, dass der Variablen y vorher ein Wert zugewiesen wurde. Hierzu werden die Funktionen

$$\mathcal{A} : \text{Stmt} \to \mathcal{P}(\text{Var})$$
$$\mathcal{D} : \text{Stmt} \times \mathcal{P}(\text{Var}) \to \text{bool}$$

definiert. $\mathcal{A}(s)$ ist die Menge der im Programm s zugewiesenen Variablen. Eine Variable heisst *zugewiesen*, wenn in dem Programm eine Zuweisung auf sie erfolgt, unabhängig davon, ob die auf der rechten Seite der Zuweisung vorkommenden Variablen initialisiert sind. $\mathcal{D}(s, A)$ ist nur dann wahr, wenn das Programm s bei Ausführung ausgehend von einem Zustand, in dem die Variablen in A initialisiert sind, nur auf initialisierte Variablen zugreift. Beachten Sie, dass $\mathcal{D}$ den Programmzustand nicht als Argument hat!

a) Definieren Sie $\mathcal{A}$ und $\mathcal{D}$. Verwenden Sie für die Definition von $\mathcal{D}$ die Funktion Vars : expr $\to \mathcal{P}(\text{Var})$, die die in einem arithmetischen oder Booleschen Ausdruck vorkommenden Variablen zurückliefert.

b) Modifizieren Sie die Big-Step-Semantik so, dass beim Benutzen einer nicht initialisierten Variable ein Fehler auftritt. Führen Sie dazu unter anderem einen Fehlerzustand **err** ein.

c) Formulieren Sie unter Verwendung von $\mathcal{D}$ eine Korrektheitsaussage für Programme in der modifizierten Big-Step-Semantik.

Aufgabe 2 (Ü) (*Eigenschaften des Operators* $\hat{R}$)

Für eine Menge $R \subseteq \mathcal{P}(M) \times M$ von Regelinstanzen (mit endlich vielen Prämissen) über der Grundmenge M ist der Operator $\hat{R} : \mathcal{P}(M) \to \mathcal{P}(M)$ definiert wie folgt:

$$\hat{R}(B) = \{y \mid \exists X \subseteq B.\ (X, y) \in R\}$$

a) Beweisen Sie den Satz:

Die Menge B ist R-abgeschlossen genau dann, wenn $\hat{R}(B) \subseteq B$.

b) Beweisen Sie die *Monotonie* von $\hat{R}$ bzgl. der Mengeninklusion, d.h.:

$$A \subseteq B \Longrightarrow \hat{R}(A) \subseteq \hat{R}(B)$$

c) Beweisen Sie:

$$\bigcup_{n\in\mathbb{N}} \hat{R}(B_n) = \hat{R}(\bigcup_{n\in\mathbb{N}} B_n)$$

für alle ω-Ketten $B_0 \subseteq B_1 \subseteq \ldots$

Aufgabe 3 (Ü) (*Anwendung des Knaster-Tarski Fixpunkt-Theorems*)

Für eine Funktion $h : M \longrightarrow N$ ist die *Erweiterung* $\tilde{h} : \mathcal{P}(M) \longrightarrow \mathcal{P}(N)$ folgendermaßen definiert für $M_1 \subseteq M$:

$$\tilde{h}(M_1) = \{n \mid \exists m \in M_1.\ h(m) = n\}$$

a) Zeigen Sie die Gültigkeit des Banach'schen Dekompositionstheorems:

Gegeben seien zwei beliebige Mengen X und Y und zwei beliebige Funktionen $f : X \longrightarrow Y$ und $g : Y \longrightarrow X$. Dann gibt es $X_1, X_2 \subseteq X$ und $Y_1, Y_2 \subseteq Y$ mit

$$X_1 \cup X_2 = X, X_1 \cap X_2 = \emptyset, Y_1 \cup Y_2 = Y, Y_1 \cap Y_2 = \emptyset,$$

so dass

$$\tilde{f}(X_1) = Y_1 \text{ und } \tilde{g}(Y_2) = X_2 \tag{1}$$

Gehen Sie dazu folgendermaßen vor:

i) Stellen Sie den Inhalt des Theorems zeichnerisch dar.

ii) Geben Sie eine Funktion $\Phi : \mathcal{P}(X) \longrightarrow \mathcal{P}(X)$ unter Zuhilfenahme der Eigenschaft (1) an, so dass X_2 ein Fixpunkt von Φ ist.

iii) Weisen Sie die Existenz eines Fixpunkts von Φ nach.

b) Zeigen Sie nun das Theorem von Schröder-Bernstein:

Gegeben seien zwei beliebige Mengen X und Y. Wenn es zwei injektive Funktionen $f : X \longrightarrow Y$ und $g : Y \longrightarrow X$ gibt, dann gibt es eine Bijektion $b : X \longrightarrow Y$.

Hinweis: Konstruieren Sie die Funktion b unter Verwendung des Theorems aus a).

Aufgabe 4 (P) (*Ausnahmebehandlung*)

Erweitern Sie die Small-Step-Semantik von WHILE um die Konstrukte **raise** und **handle**. Im Programmrahmen

`/usr/proj/semantik/prog/prolog/raise/`

finden sie einen entsprechend erweiterten Parser für WHILE. Beachten Sie, dass der Parser für s_1 **handle** s_2 die Syntax `try` s_1 `handle` s_2 `end` erwartet. Testen Sie ihre Programm und geben Sie die Prolog-Datei und die geforderten Testläufe ab.