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

Vorlesung Semantik
Sommersemester 2004
Übungsblatt 5
Besprechung am 26. Mai 2004

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. $\mathcal{D}(s, A)$ 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.

Aufbabe 2 (Ü) (*Abgeschlossene Mengen*)

Gegeben sind die Menge $N = \{1, \ldots, 12\}$ von natürlichen Zahlen, die Verknüpfung

$$n \circ m := (n \cdot m) \bmod 13$$

und die Regeln

$$3 \in A \quad \text{(Ax)} \qquad \text{und} \qquad \frac{n \in A \qquad m \in A}{n \circ m \in A} \quad \text{(R)}.$$

a) Geben Sie verschiedene Teilmengen $A \subseteq N$ an, die bezüglich der Regeln abgeschlossen sind.

b) Geben Sie die kleinste solche Menge an.

Hinweis: Benutzen Sie, dass $(N, \circ)$ ein Gruppe bildet, und bestimmen Sie die Ordnungen der Elemente von N.

Aufgabe 3 (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 4 (P) (*Compiler für Boolesche Ausdrücke*)

In dieser Aufgabe soll der Compiler von Übungsblatt 4, Aufgabe 4 auf Boolesche Ausdrücke erweitert und in Prolog implementiert werden. Die Stackmaschine wird für Boolesche Ausdrücke folgendermaßen erweitert: die Zahl 0 repräsentiert `false`, 1 steht für `true`. Die folgende Instruktion kommt hinzu:

- `Compare r`: Testet die Bedingung $t_2\, \mathtt{r}\, t_1$, wobei t_1 das oberste und t_2 das zweitoberste Stackelement ist, und $\mathtt{r} \in \{=, \leq\}$. Die obersten beiden Elemente werden entfernt und, ist die Bedingung erfüllt, so wird 1 auf den Stack geschrieben, ansonsten 0.

Erweitern Sie Ihren Compiler für Boolesche Ausdrücke. Beachten Sie dabei, dass die Stackmaschine keine Operationen für die logischen Operationen $\neg, \wedge$ und $\vee$ hat.

Implementieren Sie Stackmaschine und Compiler entsprechend dem Programmrahmen in `/usr/proj/semantik/prog/prolog/code-gen/`, und geben Sie Ihre Datei `semantik.pl` ab.