Nachklausur
Semantik von Programmiersprachen
SS98

Zu beachten:

- Es gelten dieselben Beschränkungen, wie bei der Erstklausur :-)
- Zum Bestehen der Klausur müssen Sie mindestens die Hälfte der Punkte erlangen.
- Die Klausur besteht aus 4 Abschnitten:
  - Posets und cpos
  - IMP bezogene Aufgaben
  - Hoare-Kalkül
  - Funktionale Sprachen

Viel Vergnügen beim Bearbeiten der Klausur!!

'I was gratified to be able to answer promptly, and I did. I said I didn't know.'
Mark Twain

1 Posets und cpos (10 Punkte)

Aufgabe I.1 (6 Punkte) Sei $N = \mathbb{Z}_\perp$ die flache cpo der ganzen Zahlen. Sei P die Menge aller nichtleeren Teilmengen von N mit folgender Ordnung:
A $\leq$ B gdw. $\perp \in A$ und $A \setminus \{\perp\} \subseteq B \setminus \{\perp\}$ oder $\perp \notin A$ und $A = B$.

1. Beweisen Sie, daß $(P, \leq)$ eine Poset ist. (3 Punkte)
2. Zeigen Sie, daß $s : N \to P$ definiert durch $s(x) = \{x\}$ monoton ist. (1 Punkt)
3. Zeigen Sie, daß für alle $A, B, C \in P$ gilt:

$$A \leq B \Rightarrow A \cup C \leq B \cup C. \text{ (2 Punkte)}$$

Aufgabe I.2 (1 Punkt) Seien P und Q cpo's, $S \subseteq P$ und $T \subseteq Q$. Zeigen Sie, daß $F = \{f : P \to Q \text{ stetig} \mid x \in S \Rightarrow f(x) \in T\}$ eine zulässige Teilmenge von $[P \to Q]$ ist.

Aufgabe I.3 (3 Punkte) Seien P und Q cpo's mit kleinsten Elementen und seien $f : P \to Q$ und $g : Q \to P$ stetig. Zeigen Sie $\mathit{lfp}\,(f \circ g) = f(\mathit{lfp}\,(g \circ f))$.

2 IMP and friends (8 Punkte)

Hinweis für beide Aufgaben: Kümmern Sie sich nicht um Fehlerfälle, sondern nehmen Sie (wie in der Vorlesung) an, daß alle Variablen immer einen sinnvollen Wert beinhalten, wenn auf sie zugegriffen wird.

Aufgabe II.1 (4 Punkte) In einer Erweiterung von IMP gibt es außer den üblichen Programmvariablen X, Y usw., die eine ganze Zahl speichern können, auch *Prozedurvariablen* P, Q usw., die eine Anweisung speichern können. Dazu gibt es zwei neue Arten von Anweisungen:

$P := \{C\}$ speichert die Anweisung C in der Prozedurvariablen P ab (ohne sie auszuführen);

`call` P führt die in der Prozedurvariablen P gespeicherte Anweisung aus.

Ihre Aufgabe ist es, eine *operationelle* Ergebnis-Semantik für diese neuen Anweisungen anzugeben.

Hinweis: Da diese Prozeduren parameterlos sind und es in IMP keine Deklarationen und somit keine lokalen Variablen gibt, kommen Sie weiterhin mit einem einzelnen Zustand σ aus. Dieser bildet gewöhnliche Programmvariablen nach $\mathbf{N}$ und Prozedurvariablen nach `Com`, der Menge der Anweisungen, ab.

Aufgabe II.2 (4 Punkte) In einer anderen Erweiterung von IMP gibt es außer den üblichen Programmvariablen X, Y usw., die eine ganze Zahl speichern können, auch *Zeigervariablen* Z, Z_1, Z_2 usw., die auf eine Programmvariable zeigen.

In Ausdrücken können außer gewöhnlichen Programmvariablen X (Bedeutung: die in X abgespeicherte Zahl) auch Teilausdrücke der Form $*Z$ vorkommen (Bedeutung: die Zahl, die in der Programmvariablen abgepeichert ist, auf die Z zeigt). Dazu gibt es mehrere neue Arten von Anweisungen:

$Z_1 := Z_2$ bewirkt, daß Z_1 auf die Programmvariable zeigt, auf die Z_2 zeigt;

$Z := \&X$ bewirkt, daß Z auf X zeigt;

$*Z := A$ bewirkt, daß der Wert des Ausdrucks A in der Programmvariablen abgespeichert wird, auf die Z zeigt.

Dazu gibt es natürlich nach wie vor gewöhnliche Wertzuweisungen $X := A$.

Ihre Aufgabe ist es, eine *denotationelle* Semantik für diese Erweiterung zu entwerfen. Sie brauchen dazu nur die Semantiken der Ausdrücke X und $*Z$ sowie der Anweisungen $Z_1 := Z_2$, $Z := \&X$, $*Z := A$ und $X := A$ anzugeben.

Hinweis: Benutzen Sie weiterhin einen einzelnen Zustand σ. Dieser bildet gewöhnliche Programmvariablen nach Z und Zeigervariablen nach *Loc*, der Menge der Programmvariablen, ab. Dabei bedeute $\sigma(Z) = X$, daß Z auf X zeigt.

3 Hoare-Kalkül (8 Punkte)

Aufgabe III.1 (3 Punkte) Beweisen Sie mit Hilfe der Hoare-Regeln die Gültigkeit von

$$\{X \geq 0\}\ \texttt{while}\ X > 0\ \texttt{do}\ X := X - 1\ \{X = 0\}$$

Geben Sie dabei genau an, welche Regeln Sie wie benutzen.

Aufgabe III.2 (4 Punkte) Welche der folgenden Hoare-Tripel sind gültig?

1. $\{X = 3 \wedge Y = 2\} X := X - Y \{X = 1\}$
2. $\{X = n \wedge Y = m\} X := X + Y; Y := X - Y; X := X - Y \{X = m \wedge Y = n\}$

   Unterscheiden Sie dabei jeweils die beiden Fälle, daß X und Y verschieden oder identisch sind. Begründen Sie Ihre Antworten.

Aufgabe III.3 (1 Punkt) Gilt für alle Anweisungen C und Prädikate P die Äquivalenz $wp\,(C \mid \neg P) \cong \neg wp\,(C \mid P)$? Begründen Sie Ihre Antwort!

4 Funktionale Sprachen (8 Punkte)

Aufgabe IV.1 (4 Punkte) In der funktionalen Sprache 1. Ordnung sei eine Funktion $m : i \to i \to i$ wie folgt definiert:

$m\ x\ y = \text{if } x = 0 \text{ then } 0 \text{ else } (m\ (x-1)\ y) + y.$

1. Geben Sie an, wie $m\ 2\ 3$ operationell zu einem Wert reduziert wird (call-by-value und call-by-name).
2. Ergibt für alle Ausdrücke E_1 und E_2 vom Typ i der Aufruf $m\ E_1\ E_2$ by call-by-value dasselbe wie bei call-by-name? Wenn Sie das glauben, begründen Sie es, wenn nicht, geben Sie zwei Beispielausdrücke E_1 und E_2 an, für die etwas verschiedenes herauskommt, und erklären Sie, wieso das so ist.

Aufgabe IV.2 (2 Punkte) Reduzieren Sie in der funktionalen Sprache höherer Ordnung den Ausdruck

$$(\lambda(f : i \to i \to i).\lambda(x : i).f\ x\ x)(\lambda(y : i).\lambda(z : i).\,y + z)\,3$$

operationell zu einem Wert (call-by-value und call-by-name).

Aufgabe IV.3 (2 Punkte) Für die funktionale Sprache höherer Ordnung hat jemand den vordefinierten Gleichheitsoperator auf ganzen Zahlen denotationell wie folgt definiert:

$$[\![E_1 = E_2]\!]\,\phi\,\rho = \begin{cases} \lfloor \text{true} \rfloor & \text{falls } [\![E_1]\!]\,\phi\,\rho = [\![E_2]\!]\,\phi\,\rho \\ \lfloor \text{false} \rfloor & \text{sonst} \end{cases}$$

Finden Sie heraus, wieso das so nicht geht, und korrigieren Sie den Fehler.