Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Gerwin Klein

WS 1999/2000
Übungsblatt 2
8. 11. 1999

Übungen zur Semantik von Programmiersprachen

Aufgabe 1 (H) (*Äquivalenz*) (Abgabe: Dienstag, 16.11.99)

Zwei Statements s_1 und s_2 heißen semantisch äquivalent, wenn sie gleiche Startzustände in gleiche Endzustände überführen:

$$s_1 \sim s_2 \quad :\Longleftrightarrow \quad \forall \sigma, \sigma' \in \Sigma.\ \langle s_1, \sigma\rangle \to \sigma' \iff \langle s_2, \sigma\rangle \to \sigma'$$

Zeigen Sie, daß gilt

$$(s_1; s_2); s_3 \ \sim\ s_1; (s_2; s_3)$$

Aufgabe 2 (H) (*Regelinduktion*) (Abgabe: Dienstag, 16.11.99)

Die Menge der durch drei teilbaren Zahlen **Drei** als eine Teilmenge der natürlichen Zahlen sei gegeben durch die folgenden beiden Regeln:

$$0 \in \textbf{Drei} \qquad \frac{n \in \textbf{Drei}}{n+3\ \in \textbf{Drei}}$$

Formulieren Sie die dazugehörige Induktionsregel und beweisen Sie damit die folgenden Aussagen:

(a) Die Verdoppelung einer durch drei teilbaren Zahl ergibt wieder eine durch drei teilbare Zahl.

(b) Die Addition zweier durch drei teilbaren Zahlen ergibt wieder eine durch drei teilbare Zahl.

Die für die natürlichen Zahlen üblichen Rechenregeln dürfen vorausgesetzt werden. Beachten Sie, daß die zweite zu beweisende Aussage zunächst in geeigneter Weise umgeformt werden muß, um die Formel der Regelinduktion anwenden zu können.

Aufgabe 3 (Ü) (*Semantik der While-Schleife*)

Zeigen folgende **while**-Programme dasselbe Verhalten oder nicht? (Begründung)

(a) Vergleichen Sie **while** $b_0 \vee b_1$ **do** s mit

1) **while** b_0 **do** s; **while** $b_0 \vee b_1$ **do** s 2) **while** $b_0 \vee b_1$ **do** s; **while** b_0 **do** s

(b) Vergleichen Sie **while** $b_0 \wedge b_1$ **do** s mit

3) **while** b_0 **do** s; **while** $b_0 \wedge b_1$ **do** s 4) **while** $b_0 \wedge b_1$ **do** s; **while** b_0 **do** s

Aufgabe 4 (Ü) (*Semantik der While-Schleife*)

Die aus der Vorlesung bekannten Regeln der operationalen Semantik seien in bezug auf die **while**-Schleife wie folgt modifiziert:

$$\frac{}{\langle \textbf{while}\ b\ \textbf{do}\ s, \sigma\rangle \to \sigma} \quad \text{falls} \quad B[\![b]\!]\sigma = ff$$

$$\frac{\langle \textbf{while}\ b\ \textbf{do}\ s, \sigma\rangle \to \sigma'' \qquad \langle s, \sigma''\rangle \to \sigma'}{\langle \textbf{while}\ b\ \textbf{do}\ s, \sigma\rangle \to \sigma'} \quad \text{falls} \quad B[\![b]\!]\sigma = tt$$

(a) Verändert sich dadurch die Semantik von **while**-Programmen oder nicht? Begründen Sie Ihre Antwort oder geben Sie ein Beispielprogramm an, das bezüglich der beiden Regelmengen unterschiedliche Bedeutungen besitzt.

(b) Sei $w \equiv \textbf{while}\ b\ \textbf{do}\ s$. Zeigen Sie, daß es für alle σ unter der modifizierten Semantik kein σ' gibt, sodaß $\langle w, \sigma\rangle \to \sigma'$. Betrachten Sie dazu die Form der Ableitungen.

Aufgabe 5 (P) (*Operationale Semantik in PROLOG*)

(Abgabe: Dienstag, 16.11.99, 12 Uhr an kleing@in.tum.de)

(a) In imperativen Programmiersprachen gibt es neben dem *while*-Konstrukt oft auch ein *for*-Konstrukt, bei dem eine Laufvariable bis zu einem bestimmten Wert hochgezählt wird.

Die Syntax der Sprache WHILE aus der Vorlesung sei nun um das Konstrukt

$\textbf{for}\ x\ \textbf{from}\ a_1\ \textbf{to}\ a_2\ \textbf{do}\ s$

erweitert. Geben Sie die entsprechenden Regeln für die operationelle Semantik an.

(b) Syntax und Semantik der erweiterten WHILE-Sprache sollen in PROLOG programmiert werden. Zur Vereinfachung sollen für die abstrakte Syntax statt der Infix- bzw. Mixfix-Schreibweise '$a_0 + a_t$' oder '*if* b_0 *then* s_0 *else* s_1' ausschließlich Präfixkonstrukte wie '$add(a_0, a_t)$' oder '$cond(b_0, s_0, s_1)$' verwendet werden.

Hinweise zum Starten von PROLOG auf den HP-Maschinen in der Informatikhalle und Literaturhinweise finden Sie auf den HP-Maschinen in der Datei

/usr/proj/semantik/Programme/Prolog/Natural-semantics/hinweise.txt

Eine Einführung in PROLOG wird in der Zentralübung am 9.11.99 gegeben.