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

Vorlesung Semantik
Sommersemester 2004
Semestralklausur
23. Juli 2004

Die Klausur besteht aus vier Aufgaben. Zum Bearbeiten der Klausur haben Sie 120 Minuten Zeit. Bei jeder Aufgabe ist angegeben, wieviele Punkte erreicht werden können; bei Teilaufgaben ist die Aufteilung der Punkte in Klammern angegeben. Insgesamt können 60 Punkte erreicht werden, zum Bestehen der Klausur reichen 24 Punkte aus.

Wir wünschen Ihnen viel Erfolg!

Aufgabe 1 (*Jinja*) 11 P.

In dieser Aufgabe wird eine vereinfachte Version von Jinja ohne Exceptions und mit den folgenden Modifikationen betrachtet:

- Die einzigen in der Sprache vorkommenden Werte sind Adressen. Der Nullzeiger wird durch die Adresse 0 repräsentiert, und von der Speicherverwaltung wird sichergestellt, dass dort nie ein Objekt erzeugt wird.
- Alle Methoden haben genau ein Argument.

Einige Regeln der Big-Step-Semantik sind vorgegeben.

$$P \vdash \langle \texttt{Val}\, v, s\rangle \Rightarrow \langle \texttt{Val}\, v, s\rangle \qquad \texttt{Val}\, v$$

$$l\, V = \lfloor v \rfloor \Longrightarrow P \vdash \langle \texttt{Var}\, V, (h, l)\rangle \Rightarrow \langle \texttt{Val}\, v, (h, l)\rangle \qquad \texttt{Var}\, V$$

$$\begin{array}{c}[\![P \vdash \langle e, s_0\rangle \Rightarrow \langle \texttt{Val}\, v, (h, l)\rangle;\ l' = l(V \mapsto v)]\!] \\ \Longrightarrow P \vdash \langle V \texttt{:=} e, s_0\rangle \Rightarrow \langle \texttt{Val}\, 0, (h, l')\rangle\end{array} \qquad V \texttt{:=} e$$

$$\begin{array}{c}[\![P \vdash \langle e, s_0\rangle \Rightarrow \langle \texttt{Val}\, a, (h, l)\rangle;\ h\, a = \lfloor (C, fs)\rfloor;\ fs(F, D) = \lfloor v \rfloor]\!] \\ \Longrightarrow P \vdash \langle e.F\{D\}, s_0\rangle \Rightarrow \langle \texttt{Val}\, v, (h, l)\rangle\end{array} \qquad e.F\{D\}$$

$$\begin{array}{c}[\![P \vdash \langle e, s_0\rangle \Rightarrow \langle \texttt{Val}\, v, s_1\rangle;\ v \neq 0;\ P \vdash \langle e_1, s_1\rangle \Rightarrow \langle e', s_2\rangle]\!] \\ \Longrightarrow P \vdash \langle \texttt{if}(e)\ e_1\ \texttt{else}\ e_2, s_0\rangle \Rightarrow \langle e', s_2\rangle\end{array} \qquad \texttt{if}_1$$

$$\begin{array}{c}[\![P \vdash \langle e, s_0\rangle \Rightarrow \langle \texttt{Val}\, 0, s_1\rangle;\ P \vdash \langle e_2, s_1\rangle \Rightarrow \langle e', s_2\rangle]\!] \\ \Longrightarrow P \vdash \langle \texttt{if}(e)\ e_1\ \texttt{else}\ e_2, s_0\rangle \Rightarrow \langle e', s_2\rangle\end{array} \qquad \texttt{if}_2$$

Ergänzen Sie die Regeln für Typcast $\texttt{Cast}\, C\, e$, Methodenaufruf $e_1.M(e_2)$ und die While-Schleife $\texttt{while}(e_1)e_2$.

Sie können hierbei das Prädikat $P \vdash C\, sees\, M : T_1 \to T_2 = (x, b)\, in\, D$ benutzen, welches bedeutet, dass im Programm P von der Klasse C aus die Deklaration einer Methode M mit Argumenttyp T_1, Ergebnistyp T_2, Parametername x und Methodenrumpf b in der Klasse D sichtbar ist.

Aufgabe 2 (*WHILE-Sprache*) 18 P. (5+13)

Gegeben seien in der Big-Step-Semantik der WHILE-Sprache die folgenden Regeln für die **repeat**-Schleife.

$$\frac{\langle s, \sigma\rangle \to \sigma' \qquad \mathcal{B}[\![b]\!]_{\sigma'} = \text{true}}{\langle \textbf{repeat}\ s\ \textbf{until}\ b, \sigma\rangle \to \sigma'} \qquad (\textbf{repeat}_1)$$

$$\frac{\langle s, \sigma\rangle \to \sigma' \qquad \mathcal{B}[\![b]\!]_{\sigma'} = \text{false} \qquad \langle \textbf{repeat}\ s\ \textbf{until}\ b, \sigma'\rangle \to \sigma''}{\langle \textbf{repeat}\ s\ \textbf{until}\ b, \sigma\rangle \to \sigma''} \qquad (\textbf{repeat}_2)$$

a) Geben Sie eine Transformationsfunktion T an, die Programme mit **repeat**-Schleifen in äquivalente Programme ohne **repeat**-Schleifen übersetzt.

b) Zeigen Sie für beliebige Programme

$$\langle s, \sigma\rangle \to \sigma' \quad \Longrightarrow \quad \langle T(s), \sigma\rangle \to \sigma'.$$

Beschränken Sie sich im Beweis auf die Fälle, die sich durch Zuweisung, Semikolonoperation und Repeat-Schleife ergeben.

Aufgabe 3 (*JinjaVM*) 8 P.

Gegeben ist ein JinjaVM-Programm mit den Klassen A und B. Klasse B erweitert A. In Klasse A gibt es ein Feld F vom Typ `Class` A und die folgende Methode vom Typ $[\texttt{Class}\, B, \texttt{Class}\, B] \to \texttt{Integer}$:

```
[Load 1, Store 2, Push (Intg 1), Store 2,
 Load 1, GetField F A, Goto -5, Return]
```

Die Methode wird für ein Objekt der Klasse A aufgerufen. Geben Sie den vom Bytecode-Verifier berechneten Methodentyp an.

Aufgabe 4 (*Typsystem für eine Registermaschine*) 23 P. (9+5 +9)

Gegeben ist eine Registermaschine mit den Registern $r_0, \ldots, r_{m-1}$ und dem Programmzähler p. Die Register können sowohl ganze Zahlen als auch die Booleschen Werte true und false aufnehmen. Die Maschine hat den folgenden Befehlssatz:

Instruktion	Verhalten
`Integer` c, i	$r_i := c$, wobei $c \in \mathbb{Z}$
`Sub` i, j, k	Wenn $r_i, r_j \in \mathbb{Z}$, dann $r_k := r_i - r_j$.
`Test` i, j	Wenn $r_i = 0$, dann $r_j :=$ true; wenn $r_i \in \mathbb{Z} \setminus \{0\}$, dann $r_j :=$ false.
`IfFalse` i, b	Wenn $r_i =$ false, dann $p := p + b$; wenn $r_i =$ true, dann $p := p + 1$.

Ausser bei einem bedingten Sprung wird die Programmausführung bei der nächsten Instruktion fortgesetzt. Ein Programm ist eine Liste *Is* von Instruktionen, beginnend bei der Programmadresse 0. Programmausführung endet normal, wenn der Programmzähler $|\,Is\,|$ erreicht. Enthält ein Register einen für die Ausführung einer Instruktion ungeeigneten Wert, so bleibt die Maschine stecken.

Zu Beginn der Ausführung sind die Register $r_0, \ldots, r_{n-1}$ mit den Werten $v_0, \ldots, v_{n-1}$ initialisiert, wobei $n \leq m$. Das Ergebnis wird von dem Programm in r_0 abgelegt.

a) Formulieren Sie die Übergangsrelation der Registermaschine als Small-Step-Semantik.

b) Geben Sie ein Typsystem für die Registermaschine an, in dem zwischen ganzzahligen und Booleschen Werten unterschieden wird. Definieren Sie die Begriffe *Registertyp* und *Programmtyp*, wobei der Registertyp τ jedem Register einen geeigneten Typ zuordnet, und der Programmtyp Φ jeder Instruktion eines Programms einen geeigneten Typ zuordnet.

c) Definieren Sie das Prädikat app $I\,\tau$, das angibt, ob eine Instruktion I bei gegebenem Registertyp τ anwendbar ist. Definieren Sie ebenfalls die Funktionen für Effekt eff $I\,\tau$ und die Nachfolger succs $I\,p$.