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

Vorlesung Semantik
Sommersemester 2007
Übungsblatt 5
Besprechung am 5. Juni 2007

Gegeben ist folgendes Jinja-Programm `P` mit den Klassen `X`, `Y`, `Z` und `W`:

```
class X extends Object {
  field n : Integer;
  method flip : () -> Integer = {
    Var(this).n{X} := Var(this).n{X} + Val(Intg(1)) } }

class Y extends X {
  field n : Boolean }

class Z extends Y {
  method flip : () -> Boolean = {
    Var(this).n{Y} := Var(this).n{Y} = Val(Bool(False)) } }

class W extends X {
  method flip : (i : Integer) -> Boolean = {
    Var(i) = Val(Intg(0)) } }
```

Aufgabe 1 (H) (*Deklarationsinformation*)

a) Geben Sie für alle Klassen C ein *FDTs* an, so dass $\texttt{P} \vdash C$ has-fields *FDTs* gilt.

b) Bestimmen Sie alle Tupel (C, F, T, D), für die $\texttt{P} \vdash C$ sees $F : T$ in D gilt.

c) Bestimmen Sie alle Tupel (C, M, Ts, T, B, D), für die $\texttt{P} \vdash C$ sees $M : Ts \to T = B$ in D gilt.

Aufgabe 2 (H) (*Ausführung von Jinja-Programmen*)

Reduzieren Sie den Ausdruck e bezüglich Programm `P` ausgehend vom leeren Heap und der leeren Variablenumgebung:

$$\texttt{P} \vdash \langle e, ([], [])\rangle \Rightarrow \langle e', (h', l')\rangle$$

Geben Sie jeweils die Ergebniskonfiguration $\langle e', (h', l')\rangle$ an für die folgenden Programme e:

a) `new(X); new(W)`

b) `x := new(Z); Var(x).flip()`

c) `{x : Class(X); x := new(Z); Var(x).flip()}`

Aufgabe 3 (Ü) (*Natürliche Zahlen in Jinja*)

Geben Sie eine Deklaration der Klasse `Nat` an, die die natürlichen Zahlen implementiert, ohne dabei die in Jinja vorhandenen Integer-Zahlen zu benutzen. Verwenden Sie stattdessen Zeiger. Der Nullzeiger *Null* repräsentiert die Null, ein Zeiger auf die Zahl n repräsentiert $n + 1$. Zum Beispiel ist

$$\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow (\mathit{Null})$$

die Darstellung von drei. Geben Sie Methoden für die Nachfolgeroperation `suc`, Addition `add` und den Vergleich zweier Zahlen `eq` an.

Aufgabe 4 (Ü) (*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. Hierzu müssen Sie sowohl die Semantik der arithmetischen und Booleschen Ausdrücke modifizieren, als auch die Übergangsrelation $\langle s, \sigma \rangle \to \sigma'$.

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