Institut für Informatik
und Praktische Mathematik
der Universität Kiel

Kiel, 24.04.2002

Übungen zu Organisation und Architektur von Rechnern
SS 2002
Serie 2

Aufgabe 4 (8 Punkte)

Es soll die in der Vorlesung vorgestellte SEMCD-Maschine in DrScheme implementiert werden. Definieren Sie dazu zunächst die notwendigen abstrakten Datentypen:

`(define-structure (Ap n exprlist))`	für Applikationen: $\overline{@}^{(n)}$ *exprlist*
`(define-structure (ApM n m))`	für $\overline{@}^{(n)}$-Konstruktoren auf M-Stack: $\overline{@}^{(n\|m)}$
`(define-structure (Lam n exprlist))`	für Abstraktionen: $\lambda^{(n)}$ *exprlist*
`(define-structure (Clos bindlist lam))`	für Closures: [*bindlist lam*]
`(define-structure (Bind name expr))`	für Elemente des Environments: $\langle$*name expr*$\rangle$
`(define-structure (Var name))`	für Variablen
`(define-structure (Num val))`	für Zahlen

Beispiele:

- $\overline{@}^{(3)}\ 7\ 11\ \lambda^{(3)}\ x\ y\ x \quad \equiv$
```
(make-Ap 3 (list (make-Num 7)
                 (make-Num 11)
                 (make-Lam 3 (list (make-Var "x")
                                   (make-Var "y")
                                   (make-Var "x")))))
```
- $[\langle z\ 1\rangle : \langle x\ 3\rangle : nil \quad \lambda^{(3)}\ x\ y\ x] \quad \equiv$
```
(make-Clos (list (make-Bind "z" (make-Num 1))
                 (make-Bind "x" (make-Num 3)))
           (make-Lam 3 (list (make-Var "x")
                             (make-Var "y")
                             (make-Var "x"))))
```

Die in DrScheme eingebaute Darstellung von Konstruktor-Ausdrücken ist ziemlich unübersichtlich. Definieren Sie daher für Debugging-Zwecke eine Funktion `print-expr` zur Konvertierung eines abstrakten Ausdrucks in einen String. Für die beiden o. a. Beispiele soll `print-expr` etwa folgende Ausgabe liefern:

- `"ap{3} 7 11 lam{3} x y x"`
- `"[ <z 1> : <x 3> : nil  lam{3} x y x ]"`

Aufgabe 5 (10 Punkte)

Implementieren Sie aufbauend auf Aufgabe 4 eine vereinfachte SEMCD-Maschine, die Ausdrücke lediglich vom C-Stack auf den S-Stack traversiert (Regeln (7) – (10) auf den Folien zu der Vorlesung).

Definieren Sie dazu eine Funktion `semcd`, die eine Liste mit fünf Stacks — Stacks werden ebenfalls durch Listen repräsentiert — konsumiert und eine Liste mit der neuen Stack-Konstellation zurückliefert.

Hinweis: Für Abstraktionen brauchen die Traversionsregeln *nicht* implementiert zu werden, deshalb ist in Aufgabe 4 auch keine `LamM`-Struktur definiert worden. Abstraktionen werden in der vollständigen SEMCD-Maschine ohnehin stets in eine Closure umgebaut und dann als Ganzes auf den S-Stack transportiert. Nehmen Sie also für diese Aufgabe an, daß in den zu verarbeitenden Ausdrücken keine Abstraktionen vorkommen.

Beispiel:

```
(semcd (list ()                                          ;; S is empty
             ()                                          ;; E is empty
             ()                                          ;; M is empty
             (list (make-Ap 2 (list (make-Num 7)         ;; Expression on C
                                    (make-Var "fun"))))
             ()))                                        ;; D is empty
```

liefert das Ergebnis

```
(list (list (make-Ap 2 (list (make-Var "fun")  ;; Traversed expression on S
                             (make-Num 7))))
      ()                                       ;; E is empty
      ()                                       ;; M is empty
      ()                                       ;; C is empty
      ())                                      ;; D is empty
```

.

Definieren Sie außerdem eine Funktion `print-semcd`, um SEMCD-Maschinenzustände übersichtlich ausgeben zu können. Für das obige Beispiel etwa:

Anfangszustand:
`"( nil , nil , nil , ap{2} 7 fun : nil, nil )"`

Endzustand:
`"( ap{2} fun 7 : nil , nil , nil , nil, nil )"` .

Aufgabe 6 (14 Punkte)

Erweitern Sie die Funktion `semcd` aus Aufgabe 5 zu einer vollständigen SEMCD-Maschine entsprechend den Regeln (1) – (10) auf den Folien.

Hinweis: Die Regeln (1) und (2) werden bereits vollständig durch die Regel (2a) abgedeckt.

Testen Sie Ihre Maschine unter anderem mit folgenden Beispielen:

- $\overline{@}^{(2)}\ 15\ \lambda^{(2)}\ x\ x$
- $\overline{@}^{(2)}\ 34\ \overline{@}^{(2)}\ 15\ \lambda^{(2)}\ x\ x$
- $\overline{@}^{(3)}\ b\ a\ \lambda^{(3)}\ x\ y\ x$

Aufgabe 7 (4 Punkte)

Versuchen Sie, mit Ihrer SEMCD-Maschine den folgenden Ausdruck zu berechnen:

$$\overline{@}^{(2)}\ 7\ \overline{@}^{(2)}\ 11\ \lambda^{(3)}\ x\ y\ y$$

Begründen Sie, warum es dabei zu einer Fehlermeldung kommt. Wie muß der Ausdruck modifiziert werden, damit er vollständig ausgewertet werden kann?

Aufgabe 8 (4 Punkte)

Berechnen Sie mit Ihrer SEMCD-Maschine den folgenden Ausdruck:

$$\overline{@}^{(2)}\ \lambda^{(2)}\ x\ \lambda^{(2)}\ y\ x\ \lambda^{(2)}\ k\ \overline{@}^{(2)}\ k\ \overline{@}^{(2)}\ k\ \lambda^{(2)}\ x\ \lambda^{(2)}\ y\ \lambda^{(2)}\ z\ \overline{@}^{(2)}\ \overline{@}^{(2)}\ z\ y\ \overline{@}^{(2)}\ z\ x$$

a) Welches Ergebnis erhalten Sie?

b) Wie läßt sich die Funktionalität des Ergebnisses ermitteln?

Abgabe: Di., 14.05.2002 in den Übungen.