

Übungen zum Schemekurs

Aufgabe 23. Man schreibe eine nullstellige Prozedur `load-int`, die den Interpreter lädt und es gestattet, Ausdrücke wie in Petite-Scheme einzugeben und auszuwerten. Dabei soll `load-int` bevor ein Ausdruck evaluiert wird mit der Prozedur `expr?` von Aufgabe 19 testen, ob der eingegebene Ausdruck ein Scheme-Ausdruck ist. Der Interpreter soll verlassen werden können durch Eingabe von `(exit-int)`.

Hinweis: Eine sehr einfache `load-int` Prozedur sieht wie folgt aus:

```
(define (rep-loop)
  (newline)
  (display "$> ")      ; print a prompt
  (write (evl (read))) ; read expr., pass to eval, write result
  (rep-loop))
```

Lösung.

```
(define (load-int)
  (newline)
  (display "$> ")
  (let* ((input (read)))
    (if (and (list? input) (eq? (car input) 'exit-int))
        ()
        (begin
          (if (expr? input)
              (write (evl input))
              (display "not a Scheme expression"))
          (load-int))))))
```

Aufgabe 24. Petit-Scheme hat eine eingebaute Funktion (nämlich `trace`), mit der man verfolgen kann, welche Prozeduren mit welchen Argumenten aufgerufen werden. Man modifiziere die Prozedur `evl` dahingehend, dass eine ähnliche Funktionsweise im Interpreter bereitgestellt wird. (Man beachte, dass im Interpreter schon eine Funktion `trace?` implementiert wurde, die Informationen über die Speicher-verwaltung ausgibt.)

Lösung.

Um jeden Prozeduraufruf zu verfolgen (was das in Petite-Scheme eingebaute `trace` macht), ändere man

```
(app v0 vs)
```

in der `evl`-Funktion zu

```
(begin
  (tracing (cons w0 vs))
  (app v0 vs))
```

und füge zum Code hinzu

```
(define tracing? #f)
```

```
(define (tracing ws)
  (if tracing? (begin (display (map user-display ws)) (newline))))
```

Beispiel:

```
(define tracing? #t)
(ev1 '(define fib (lambda (n)
                    (if (<= n 1)
                        1
                        (+ (fib (- n 1)) (fib (- n 2)))))))
```

```
(ev1 '(fib 3))
```

```
(fib 3)
(<= 3 1)
(- 3 1)
(fib 2)
(<= 2 1)
(- 2 1)
(fib 1)
(<= 1 1)
(- 2 2)
(fib 0)
(<= 0 1)
(+ 1 1)
(- 3 2)
(fib 1)
(<= 1 1)
(+ 2 1)
3
```

Abgabe. Freitag, den 15. Oktober 2004, vor der Vorlesung oder (besser) per Email
an urban@mathematik.uni-muenchen.de