Institut für Informatik
und Praktische Mathematik
der Universität Kiel

Kiel, 17.4.2002

Übungen zu Organisation und Architektur von Rechnern
SS 2002
Serie 1

Aufgabe 1 (6 Punkte)

Die natürlichen Zahlen lassen sich als Schachtelung der Konstruktoren `zero()` und `succ( x)` darstellen. Dabei repräsentiert `zero()` die Zahl 0 und `succ( x)` dient als Repräsentation für den Nachfolger der von `x` repräsentierten Zahl, d.h.

$$0 \equiv \texttt{zero()}$$
$$1 \equiv \texttt{succ(zero())}$$
$$2 \equiv \texttt{succ(succ(zero()))}$$
$$\ldots$$

Implementieren Sie diese Zahlendarstellung in DrScheme mit Hilfe zweier abstrakter Datentypen `zero` und `succ`. Führen Sie diese mit Hilfe der in DrScheme unterstützten `define-struct` Form (siehe Link auf der Web-Seite zur Übung) ein:

```
(define-struct zero ())
(define-struct succ (x))
```

Definieren Sie zwei Funktionen `num2ast` und `ast2num` zur Konversion von Integer-Darstellungen in die entsprechenden abstrakten Darstellungen und zurück.

Aufgabe 2 (6 Punkte)

Erweitern Sie die abstrakte Repräsentation von natürlichen Zahlen aus Aufgabe 1 um Konstrukte zur Darstellung der Addition sowie der Multiplikation:

```
(define-struct add (arg1 arg2))
(define-struct mul (arg1 arg2))
```

Schreiben Sie eine Funktion `eval`, die Schachtelungen dieser Konstrukte mit den in Aufgabe 1 vorgestellten Konstrukten derart transformiert, daß das Resultat ausschliesslich aus den Konstruktoren `zero()` und `succ( x)` besteht. Führen Sie diese Transformationen ausschließlich auf der abstrakten Darstellung durch. Auf die Verwendung der Addition/Multiplikation von Integer-Zahlen ist zu verzichten! Beachten Sie dabei, dass `add`- bzw. `mul`-Konstrukte auch innerhalb des `succ`-Konstruktes auftreten können.

Bsp.: `(make-succ (eval (make-add (make-zero) (make-succ (make-zero)))))` entspricht `(make-succ (make-succ (make-zero)))`.

Aufgabe 3 (8 Punkte)

Reimplementieren Sie den abstrakten Evaluierer aus Aufgabe 2 mit Hilfe der in der Bibliothek "`match.ss`" zur Verfügung gestellten Pattern-Matching-Konstrukte sowie der in "`defstru.ss`" bereitgestellten erweiterten `define-structure` Form anstelle der `define-struct` Form.

Importieren Sie diese Bibliotheken mittels

```
(require-library "match.ss")
(require-library "defstru.ss")
```

am Beginn Ihres Programmes.

Abgabe: Di., 30.4.2002 in den Übungen.