Prof. Dr. Manfred Schmidt-Schauß
Künstliche Intelligenz/Softwaretechnologie

Fachbereich Informatik (20)
Johann Wolfgang Goethe-Universität Frankfurt am Main

Einführung in das funktionale Programmieren

Wintersemester 1995/96

Übungsblatt Nr. 2

> We chose pure LISP as a model for our language because pure LISP was designed as a mathematical language whose formulas could easily be represented within computers. Because of its mathematical nature (...) pure LISP is considered a 'toy' programming language.
>
> Boyer, Moore: A Computational Logic, 1979

Aufgabe 1
In der Vorlesung wurden Zahlen durch Funktionen dargestellt. Die Zahl 0 ist die Funktion:

```
zero x y = y
```

Die Zahl 1 ist dann der Nachfolger von 0, die 2 der Nachfolger von 1 usw. . Wir definieren also

```
eins = succ zero
zwei = succ eins = succ (succ zero)
...
```

Entsprechend ist die Zahl 0 der Vorgänger von 1, die 1 der Vorgänger von 2 usw. .

a) Werten Sie zunächst die Ausdrücke

```
iszero drei     (= zero?)
iszero zero
plus zwei drei
pred drei
times zwei zwei
```

per Hand in Normalordnung aus und verifizieren Sie Ihre Ergebnisse, indem Sie die entsprechenden Ausdrücke in Gofer auswerten lassen. Benutzen Sie für die einzelnen Funktionen die Definitionen aus dem Script.

b) Werten Sie die Ausdrücke

```
minus drei eins
minus drei zwei
```

in Normalordnung aus. Was passiert im zweiten Ausdruck, wenn Sie diesen in Gofer auswerten lassen?

Programmieraufgabe (wird fortgesetzt)
In Gofer können wir algebraische Datentypen definieren. Wir wollen CORE-FP-Ausdrücke in Gofer darstellen. Hierzu definieren wir einen sehr einfachen Datentyp:

```
data Core-Fp
 =  VarConst String
   |App Core-Fp Core-Fp
```

- Schreiben Sie eine Goferfunktion `ppcorefp`, die als Eingabe einen Ausdruck des Typs `Core-Fp` hat und als Ausgabe ein String, der diesen Ausdruck textuell darstellt.
  Beispiel:
  `(ppcorefp (App (App (VarConst "+") (VarConst "x"))(VarConst "1"))`
  $\Rightarrow$ `(+ x 1)`

- Schreiben Sie eine Goferfunktion `nextredex`, die als Argumente einen Ausdruck des Typs `Core-Fp` und eine Liste von Paaren der Art `(String,Int)` hat. Die Liste von Paaren gibt von jeder als String dargestellten Funktion ihre maximale Stelligkeit an. Ausgabe soll eine Darstellung des nächsten in der normalen Reihenfolge auszuwertenden Redex des eingegebenen Core-Fp Ausdrucks sein.

Aufgabe 2
Wie wir in der Vorlesung gesehen haben, braucht man in CORE-FP gar nicht die Möglichkeit algebraische Typen mit Konstruktoren darzustellen. Man kann dieses voll funktional machen. Eine Implementierung des obigen Goferdatentyps `Core-Fp` ist:

```
true  x y = x
false x y = y

iff x y z = x y z

app x y z n = n (z x y) true
varconst x z n = n x false

isapplication n = n true false

functionofapp  x     = iff (isapplication x) (x true true) bot
argofapp       x     = iff (isapplication x) (x false true) bot

varname x     = iff (isapplication x) bot (x bot true)

bot = bot
```

a) Beweisen Sie, daß obiges CORE-FP Programm dieselbe Funktionalität wie folgendes Goferprogramm hat:

```
isapplication (App _ _) = True
isapplication (VarConst _) = False
```

```
functionofapp (App f _) = f
argofapp (App _ arg) = arg

varname x (VarConst v) = v
```

b) Versuchen Sie das CORE-FP-Programm aus a) in Gofer einzuladen. Was passiert?

c) Übersetzen Sie das CORE-FP-Programm aus a) in Scheme.
Beachten Sie dabei, daß Sie wegen der applikativen Auswertung von Scheme kein eigenes *if* definieren können.