

Übungen zur „Deskriptiven Programmierung“

Blatt 3

Aufgabe 3. *Terme erster Stufe* können in Haskell wie folgt repräsentiert werden.

```
type Symbol = String
data Term a = Var a | Fun Symbol [Term a]
```

Ein Term ist entweder eine Variable oder ein zusammengesetzter Term, ein Funktionssymbol angewendet auf eine Folge von Termen. Wir repräsentieren Funktionssymbole durch Zeichenketten, legen uns aber bei der Darstellung von Variablen nicht fest (der Datentyp *Term* ist entsprechend parametrisiert).

Implementiere die folgenden Funktionen.

```
isVar      :: Term a → Bool
isGround  :: Term a → Bool
```

Die Funktion *isVar* testet, ob es sich bei dem Argument um eine einzelne Variable handelt; *isGround* überprüft, ob das Argument variablenfrei ist (ein Term ohne Variablen heißt auch Grundterm).

```
vars      :: Term a → [a]
occurs    :: (Eq a) ⇒ a → Term a → Bool
```

Die Funktion *vars* bestimmt die in einem Term enthaltenen Variablen; *occurs* testet, ob die übergebene Variable in dem Term enthalten ist.

Aufgabe 4. Eine *Substitution* ist eine Abbildung (eine Funktion) von Variablen auf Terme: $a \rightarrow \text{Term } b$. Definiere eine Funktion, die eine Substitution auf einen Term anwendet.

```
apply :: (a → Term b) → (Term a → Term b)
```

Definiere eine Funktion, die zwei Substitutionen komponiert.

```
(&) :: (a → Term b) → (b → Term c) → (a → Term c)
```

Zeige die folgenden Eigenschaften für beliebige Substitutionen s , t und u .

```
Var & s    = s
s & Var    = s
s & (t & u) = (s & t) & u
```