Dr. Steffen Jost
Leah Neukirchen

Ludwig-Maximilians-Universität München
Institut für Informatik
12. Juni 2018

08. Übung zur Vorlesung
Programmierung und Modellierung

A8-1 ***Typsignaturen*** Lösen Sie diese Aufgabe mit Papier und Bleistift! Geben Sie zu jeder der folgenden Deklaration eine möglichst allgemeine Signatur an. Begründen Sie Ihre Antwort informell!

Überlegen Sie sich dazu, welche Argumente jeweils auftreten und wie diese verwendet werden, z.B. welche Funktion auf welches Argument angewendet wird. Falls Sie eine der verwendeten Funktion nicht kennen, schlagen Sie diese in den Vorlesungsfolien oder der Dokumentation der Standardbibliothek nach. Verwenden Sie GHCI höchstens im Nachhinein, um Ihre Antwort zu kontrollieren. Numerische Typklassen müssen Sie zur Vereinfachung nicht angeben, verwenden Sie einfach einen konkreten Typ wie `Int` oder `Double`.

a) `gaa xy z vw = if (read z == xy) then minBound else vw`

Lösungsbeispiel: Wir sehen als erstes, dass es sich um eine Funktion mit drei Argumenten handelt, deren Typ wir bestimmen müssen, sowie den Ergebnistyp der Funktion. Nennen wir diesen Typ vorläufig einmal $a \to b \to c \to d$, also `xy` $:: a$, `z` $:: b$ und `vw` $:: c$.

Argument `xy` wird in einem Vergleich verwendet, also muss a in der Typklasse `Eq` sein.

`z` ist ein Argument für die Funktion `read` und damit ist $b =$ `String`. Da das Ergebnis aber mit `xy` verglichen wird, muss es den gleichen Typ haben, und wir wissen also, dass a auch in der Typklasse `Read` sein muss, denn es wurde ja ein Wert dieses Typs geparsed.

`vw` wird nur als Ergebnis verwendet, daraus können wir $c = d$ schließen. Da die beiden Zweige eines Konditionals den gleichen Typ haben müssen und im `then`-Zweig der Wert `minBound` zurückgegeben wird, muss dieser Typ auch noch in der Klasses `Bounded` sein.

Wir haben also `gaa :: (Eq a, Read a, Bounded c) => a -> String -> c -> c`

Hinweis: Namen von Typvariablen und Reihenfolge der Class Constraints ist unbedeutend. `gaa :: (Read y2, Bounded zz, Eq y2) => y2 -> [Char] -> zz -> zz` wäre z.B. eine äquivalente Typsignatur.

b)
```
guu _ (h1:h2:t) = [h2]
guu x _         = show x
```

c)
```
foo a b []    e f = show a
foo a b (c:d) e f | a==c, read b  = e ++ e
                  | a/=c          = show d
                  | otherwise     = show e
```

d)
```
bar a b c d | b >= c    = bar b a d c
            | otherwise = a==d
```

e)
```
gnarf [] _ = mempty
gnarf x  y = y `mappend` x `gnarf` y
```

A8-2 *Freie Variablen und Substitution*

a) Berechnen Sie jeweils die Menge der freien Variablen folgender Terme:

i) `(p q) r`

ii) `(\a -> b a) (\c -> (d c) e)`

iii) `(\x -> x) (\y -> y) (\z -> z)`

iv) `(\u -> v) (\v -> u) (\w -> w)`

v) `(\f -> (\g -> (\h -> (h f) i))) (g j)`

b) Berechnen Sie jeweils folgende Termsubstitutionen:

i) `(\a -> b a)[x/a, y/b]`

ii) `((\x -> (\y -> x (y z))) x)[a/x,b/y,(\c -> d)/z]`

iii) `p (q r)[q/r,p/t,s/q]`

iv) `(\u -> (\v -> u w))[(u v)/w]`

Tipp: Kontrollieren Sie Ihre Antworten mithilfe Ihrer Lösung zur **??**.

A8-3 *Typherleitung I*

a) Erstellen Sie eine Typherleitung in Baum-Notation für folgendes Typurteil:

$$\{\,\} \vdash (\backslash x \texttt{ -> } (\backslash y \texttt{ -> } (\backslash z \texttt{ -> } x\ z\ y))) :: (\alpha \to \beta \to \delta) \to \beta \to \alpha \to \delta$$

Zusatzfrage: Welchen Namen hat diese Funktion in der Standardbibliothek von Haskell?

b) Erstellen Sie eine Typherleitung in linearer Notation für folgendes Typurteil:

$$\{x{::}\texttt{Bool}, y{::}\texttt{Double} \to \texttt{Int}\} \vdash (\backslash z \texttt{ -> } z\ x\ 4\ (y\ 7)) :: (\texttt{Bool} \to \texttt{Int} \to \texttt{Int} \to \beta) \to \beta$$

Hinweis: Beachten Sie dabei die übliche Klammerkonventionen für Funktionstypen und Funktionsanwendung!

c) Beweisen Sie durch eine saubere Typherleitung in der Notation Ihrer Wahl, dass die folgende Haskell Funktion den behaupteten Typ hat:

```
twice :: (a -> a) -> a -> a
twice f x = f (f x)
```

Hinweis: Da das in der Vorlesung behandelte Typsystem nicht mit benannten Funktionen umgehen kann, müssen wir zuerst den Funktionsrumpf in eine äquivalenten Term übersetzen, welcher eine anonyme Funktionsdefinition mit Lambda benutzt.

H8-1 ***Substitution*** (2 Punkte; Datei `H8-1.hs` als Lösung abgeben)
Gegeben ist folgende Datentypdeklaration zur Repräsentation von Termen:

```
data Term = Var Char | Const Int | App Term Term | Abs Char Term
```

a) Schreiben Sie eine Funktion `freeVars :: Term -> [Char]` welche die freien Variablen eines Terms effizient berechnet. *Hinweis:* Die Aufgabe ist sehr einfach, wenn Sie den Unterschied zwischen freien und gebundenen Variablen verstanden haben, siehe dazu auch Folien 8.10

b) Implementieren Sie die Substitution von Folie 8.9 effizient als Funktion `subst :: (Char, Term) -> Term -> Term`. `subst ('x', t1) t2` berechnet den Term, den man erhält wenn man in `t2` alle freien Vorkommen von `Var 'x'` durch `t1` ersetzt.

Verwenden Sie zur Vereinfachung folgende partielle Funktion zur Erzeugung frischer Variablen:

```
genFreshV :: [Char] -> Char
genFreshV vs = head $ filter (\c -> not $ c `elem` vs) ['a'..'z']
```

H8-2 ***Typherleitung II*** (2 Punkte; Abgabe: `H8-2.txt` oder `H8-2.pdf`)
Erstellen Sie eine Typherleitung für folgendes Typurteil:

$$\{f{::}(\alpha \to \alpha) \to \beta \to \beta\} \vdash (\backslash x \texttt{ -> } f\ x)\ (\backslash y \texttt{ -> } y) :: \beta \to \beta$$

Sie dürfen sich aussuchen, ob Sie die Herleitung in Baum-Notation oder in linearer Notation verfassen.

H8-3 ***Unifikation*** (2 Punkte; Abgabe: `H8-3.txt` oder `H8-3.pdf`)

a) Berechnen Sie den allgemeinsten Unifikator für folgende Menge von Typgleichungen:

$$\{\texttt{Int} \to \alpha = \beta \to \texttt{Bool} \to \beta, \gamma \to \texttt{Int} = \alpha\}$$

b) Gegeben ist folgende Menge von Typgleichungen:

$$\alpha \to \beta = \delta \to \gamma, \beta \to \delta = \theta \to \eta, \eta = \alpha$$

Geben Sie eine nicht-leere Substitution an, welche alle Variablen durch konkrete Typen ersetzt, aber welche kein Unifikator für die Gleichungsmenge ist.

c) Gegeben ist folgende Menge von Typgleichungen:

$$\eta \to \beta = \delta \to \theta, \alpha = \eta, \alpha \to \beta = \delta \to \gamma$$

Geben Sie einen Unifikator für die Gleichungsmenge an, welche nicht der allgemeinste Unifikator ist.

Abgabe: Lösungen zu den Hausaufgaben können bis Samstag, den 23.6.18, mit UniWorX nur als `.zip` abgegeben werden. Abschreiben bei den Hausaufgaben gilt als Betrug und kann zum Ausschluss von der Klausur zur Vorlesung führen. Bis zu 3 Studierende können gemeinsam als Gruppe abgeben. Bitte beachten Sie auch die Hinweise zum Übungsbetrieb auf der Vorlesungshomepage (`www.tcs.ifi.lmu.de/lehre/ss-2018/promo/`).