Prof. Martin Hofmann, PhD
Dr. Steffen Jost

Ludwig-Maximilians-Universität München
Institut für Informatik
23. Juni 2015

9. Übung zur Vorlesung

Programmierung und Modellierung

Hinweise:

- Einmalige Raumänderung für die Übungen am Freitag, den 3.7.2015:
  10:00h→E216, 12:00h→E216, 14:00h→A014 im gleichen Gebäude.
- Donnerstag, 9.7., 17:00, B201: Freiwillige Fragestunde der ProMo-Tutoren!

A9-1 *Hello World* Kreieren Sie mit GHC (nicht GHCI) eine ausführbare Datei, welche nach dem Start eine Begrüßung ausgibt und den Benutzer dazu auffordert, in je eine Zeile nacheinander zuerst ein Lieblingstier und dann eine Lieblingseigenschaft einzugeben. Danach soll das Programm noch einen einzelnen String ausgeben, welcher zuerst die Lieblingseigenschaft und dann das Lieblingstier wiedergibt:

```
> ghc helloTier.hs
...
> ./helloTier
Hi! Gib bitte zuerst Dein Lieblingstier und dann
in die nächste Zeile Deine Lieblingseigenschaft ein:
Schildkröte
faule
Psst, willst Du faule Schildkröte kaufen?
```

Hinweis: Das Beispiel zeigt eine Linux-Konsole. Je nach Betriebssystem kann der Aufruf einer ausführbaren Datei abweichen. Die Benutzereingabe waren “Schildkröte” und “faule” in der drittletzten und vorletzten Zeile, der Rest wurde durch das Programm ausgegeben.

A9-2 *DO-Notation* Versuchen Sie, diese Aufgabe mit Papier und Bleistift zu lösen. Verwenden Sie GHC oder GHCI erst, wenn Sie nicht mehr weiter wissen. Was gibt das folgende Programm am Bildschirm aus? Wie oft wartet das Programm auf eine Benutzereingabe?

Hinweis: Sie dürfen sich selbst ausdenken, was der Benutzer bei jeder Eingabeaufforderung eingibt – alle Eingaben sollten jedoch verschieden sein. Die Aktion `hSetBuffering` (siehe Folie 09-30) können Sie hier ignorieren; diese sorgt nur dafür, dass das Programm auf allen Betriebssystemen gleich funktioniert.

```
import System.IO
main = do hSetBuffering stdout NoBuffering
          putStr "A: "
          a2 <- getLine
          b1 <- putStr "B: "
          let b2 = getLine
          let c1 = putStr "C: "
          c2 <- getLine
          putStr "D: "
          b2 <- b2
          putStrLn $ "A="++a2++" B="++b2++" C="++c2
```

A9-3 *Der richtige Kontext* Berechnen Sie mit Papier & Bleistift für jedes der folgenden Typisierungsurteile einen möglichst *allgemeinen* und *minimalen* Kontext Γ, so dass das entsprechende Typurteil wahr wird. Typklassen ignorieren wir in dieser Aufgabe.

Beispiel: Das Typurteil $\Gamma \vdash x + 1.0 :: \texttt{Double}$ ist z.B. für den Kontext $\Gamma = \{x{::}\texttt{Double}\}$ wahr (welcher auch minimal ist, d.h. keine unnötigen Variablen enthält), nicht aber jedoch für den Kontext $\Gamma = \{y{::}\texttt{Double}, x{::}\texttt{Int}\}$.

a) $\Gamma_a \vdash$ `if x then "Akzeptiert!" else f y` $::$ `String`

b) $\Gamma_b \vdash$ `\y -> \z -> x z (y z)` $:: (\alpha \to \beta) \to \alpha \to \gamma$

c) $\Gamma_c \vdash$ `\x -> if x then \y -> y x else \z -> 1.0` $::$ `Bool` $\to$ (`Bool` $\to$ `Double`) $\to$ `Double`

GHCI's Typinferenz kann diese Aufgaben auch lösen. *Nachdem* Sie die Aufgabe von Hand berechnet haben können Sie Ihr Ergebnis mit GHCI überprüfen. Was müssen Sie dazu jedoch mit dem Programmausdruck vorher noch tun?

H9-1 ***Hello Again*** (2 Punkte; Datei `H9-1.hs` als Lösung abgeben)

Ändern Sie Ihre Lösung zu Aufgabe A9-1 wie folgt ab:

a) Falls beide Eingaben leer waren, soll als Antwort nur der String `"Spielverderber!"` ausgeben werden, und danach soll das Programm wieder automatisch von vorne beginnen.

b) Falls nur die Eingabe für das Tier leer war, so beginnt das Programm ebenfalls von vorne, aber merkt sich heimlich die eingegebene Lieblingseigenschaft. Wenn danach mal Tier und Eigenschaft komplett eingegeben werden, wird die komplette Liste aller zuvor eingegeben Eigenschaften ausgeben.

Beispiel:

```
> ./helloTier3
Hi! Gib bitte zuerst Dein Lieblingstier und dann
in die nächste Zeile Deine Lieblingseigenschaft ein:


Spielverderber!
Hi! Gib bitte zuerst Dein Lieblingstier und dann
in die nächste Zeile Deine Lieblingseigenschaft ein:

tolle
Tier eingeben!
Hi! Gib bitte zuerst Dein Lieblingstier und dann
in die nächste Zeile Deine Lieblingseigenschaft ein:

schnelle
Tier eingeben!
Hi! Gib bitte zuerst Dein Lieblingstier und dann
in die nächste Zeile Deine Lieblingseigenschaft ein:
Kröte
grüne
Psst, willst Du grüne schnelle tolle Kröte kaufen?
```

Hinweis: Für die erste Teilaufgabe könnte Ihnen Folie 09-20 die notwendige Inspiration liefern. Für die zweite Teilaufgabe muss man vielleicht etwas nachdenken. Wir verraten nur so viel: die Lösung benötigt keineswegs irgendwelche monadischen Tricks; es reicht ein gewöhnlicher funktionaler Akkumulator.

H9-2 ***Typregel für Case*** (3 Punkte; Abgabeformat: Text oder PDF)

Erstellen Sie eine neue Typregel für Case-Ausdrücke! Zur Vereinfachung der Aufgabe hat unser Case-Ausdruck immer genau zwei Fälle, und die Patterns sind auch festgelegt auf `Nothing` im ersten Mustervergleich und `Just x` im zweiten. Der Ausdruck hat also immer die Form `case` e_1 `of {Nothing ->` e_2`; Just` x `->` e_3`}` wobei die e_1, e_2, e_3 und x hier Metavariablen sind. Überlegen Sie sich, welche Anforderungen Haskell an diesen Ausdruck stellt und formalisieren Sie diese durch eine Typregel!

H9-3 ***Typfehler*** (4 Punkte; Abgabeformat: Text oder PDF)

Geben Sie eine Herleitung (Baum- oder lineare Notation) für jedes der folgenden Typurteile an, falls möglich. Anderfalls begründen Sie, warum eine Typherleitung nicht möglich ist.

a) $\{f{::}\gamma \to \alpha \to \beta,\ g{::}\beta \to \gamma,\ x{::}\beta\} \vdash f\ g\ x :: \alpha \to \beta$

b) $\{f{::}\alpha \to$ `Bool`$,\ z{::}\beta\} \vdash$ `\`y `->` `\`x `->` `if` $f\ x$ `then` z `else` $y\ z :: (\beta \to \beta) \to \alpha \to \beta$

c) $\{\,\} \vdash$ `\`x `->` (`\`f `->` $f\ x\ x$) $:: \alpha \to (\alpha \to \alpha) \to \alpha$

Abgabe: Lösungen zu den Hausaufgaben können bis Dienstag, den 30.06.2015, 11:00 Uhr mit UniworX abgegeben werden. Abschreiben ist verboten!