Prof. Martin Hofmann, PhD
Dr. Steffen Jost

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

8. Musterlösung zur Vorlesung

Programmierung und Modellierung

A8-1 ***Unifikation I*** Berechnen Sie jeweils den allgemeinsten Unifikator für die folgenden Typgleichungen mit Robinson's Unifikationsalgorithmus, Folie 08-48, falls möglich:

a) $\{\texttt{Int} \to (\alpha \to \beta) = \alpha \to (\beta \to \texttt{Int})\}$ $[\texttt{Int}/\alpha, \texttt{Int}/\beta]$

b) $\{\alpha \to (\alpha \to \texttt{Int}) = (\texttt{Int} \to \beta) \to \beta\}$ Typfehler: Zirkulärer Typ $(\texttt{Int} \to \beta) \to \texttt{Int} = \beta$

c) $\{\alpha \to (\alpha \to \texttt{Int}) = (\texttt{Int} \to \beta) \to \gamma\}$ $[(\texttt{Int} \to \beta)/\alpha, ((\texttt{Int} \to \beta) \to \texttt{Int})/\gamma]$

A8-2 ***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?

LÖSUNGSVORSCHLAG:

Zur Vermeidung von Flüchtigkeitsfehlern empfiehlt es sich, hier erst einmal alle impliziten Klammern explizit einzufügen, siehe auch Folie 01-47.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x{::}\alpha \to \beta \to \delta \in \Delta}{\Delta \vdash x :: \alpha \to (\beta \to \delta)}\textsc{Var} \qquad \dfrac{z{::}\alpha \in \Delta}{\Delta \vdash z :: \alpha}\textsc{Var}}{\Delta \vdash x\ z :: \beta \to \delta}\textsc{App} \qquad \dfrac{y{::}\beta \in \{x{::}\alpha \to \beta \to \delta, y{::}\beta, z{::}\alpha\}}{\{x{::}\alpha \to \beta \to \delta, y{::}\beta, z{::}\alpha\} \vdash y :: \beta}\textsc{Var}}{\{x{::}\alpha \to \beta \to \delta, y{::}\beta, z{::}\alpha\} \vdash (x\ z)\ y :: \delta}\textsc{App}}{\{x{::}\alpha \to \beta \to \delta, y{::}\beta\} \vdash \backslash z\ \texttt{->}\ (x\ z)\ y :: \alpha \to \delta}\textsc{Abs}}{\{x{::}\alpha \to \beta \to \delta\} \vdash \backslash y\ \texttt{->}\ (\backslash z\ \texttt{->}\ (x\ z)\ y) :: \beta \to \alpha \to \delta}\textsc{Abs}}{\{\,\} \vdash \backslash x\ \texttt{->}\ (\backslash y\ \texttt{->}\ (\backslash z\ \texttt{->}\ (x\ z)\ y)) :: (\alpha \to (\beta \to \delta)) \to \beta \to \alpha \to \delta}\textsc{Abs}$$

wobei wir aus Platzgründen die Abkürzung $\Delta := \{x{::}\alpha \to \beta \to \delta, y{::}\beta, z{::}\alpha\}$ für einen Typkontext definieren und im oberen linken Teilbaum verwenden.

Der Programmausdruck beschreibt die Funktion `flip`.

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!

LÖSUNGSVORSCHLAG:

Aus Platzgründen definieren wir

$$\Gamma := \{x :: \texttt{Bool}, y :: \texttt{Double} \to \texttt{Int}, z :: \texttt{Bool} \to (\texttt{Int} \to (\texttt{Int} \to \beta))\}$$

in der folgenden Typherleitung im linearen Stil – natürlich können wir diese Abkürzung erst definieren, wenn wir später während der Konstruktion der Typherleitung wissen was wir wirklich brauchen, d.h. bei einer Herleitung mit Papier & Bleistift müssten die Abkürzung unten auf dem Blatt niederschreiben (oder anfangs oben etwas Platz lassen). Wir beginnen damit, in dem wir Programmausdruck und Typausdruck vollständig klammern – das ist nicht notwendig, macht aber klarer was zu tun ist.

$\{x :: \texttt{Bool}, y :: \texttt{Double} \to \texttt{Int}\} \vdash \backslash z \texttt{ -> } ((z\ x)\ 4)\ (y\ 7) :: (\texttt{Bool} \to (\texttt{Int} \to (\texttt{Int} \to \beta))) \to \beta$	ABS(2)	(1)
$\Gamma \vdash ((z\ x)\ 4)\ (y\ 7) :: \beta$	APP(3, 4)	(2)
$\Gamma \vdash (z\ x)\ 4 :: \texttt{Int} \to \beta$	APP(5, 6)	(3)
$\Gamma \vdash y\ 7 :: \texttt{Int}$	APP(9, 10)	(4)
$\Gamma \vdash z\ x :: \texttt{Int} \to (\texttt{Int} \to \beta)$	APP(7, 8)	(5)
$\Gamma \vdash 4 :: \texttt{Int}$	CONST	(6)
$\Gamma \vdash z :: \texttt{Bool} \to (\texttt{Int} \to (\texttt{Int} \to \beta))$	VAR	(7)
$\Gamma \vdash x :: \texttt{Bool}$	VAR	(8)
$\Gamma \vdash y :: \texttt{Double} \to \texttt{Int}$	VAR	(9)
$\Gamma \vdash 7 :: \texttt{Double}$	CONS	(10)

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

```haskell
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.

LÖSUNGSVORSCHLAG:

Wir müssen den Funktionsrumpf in einen geschlossenen Programmausdruck verwandeln. Die beiden Argumente der Funktion `twice` müssen also mit Lambdas gebunden werden. Dies können wir auch in Haskell so hinschreiben:

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

Damit können wir nun ganz normal unsere Typherleitung basteln. Wir wählen hier die Baum-Notation, da diese sich nach unserer Ansicht vielleicht etwas schwieriger hinschreiben läßt, dafür aber leichter zu lesen ist, so fern der Baum auf eine Seite passt.

$$
\dfrac{
  \dfrac{
    \dfrac{f \in \{f{::}\alpha \to \alpha, x{::}\alpha\}}{\{f{::}\alpha \to \alpha, x{::}\alpha\} \vdash f :: \alpha \to \alpha}\,\textsc{Var}
    \qquad
    \dfrac{
      \dfrac{f \in \{f{::}\alpha \to \alpha, x{::}\alpha\}}{\{f{::}\alpha \to \alpha, x{::}\alpha\} \vdash f :: \alpha \to \alpha}\,\textsc{Var}
      \quad
      \dfrac{x \in \{f{::}\alpha \to \alpha, x{::}\alpha\}}{\{f{::}\alpha \to \alpha, x{::}\alpha\} \vdash x :: \alpha}\,\textsc{Var}
    }{\{f{::}\alpha \to \alpha, x{::}\alpha\} \vdash f\ x :: \alpha}\,\textsc{App}
  }{
    \dfrac{\{f{::}\alpha \to \alpha, x{::}\alpha\} \vdash f\ (f\ x) :: \alpha}{\{f{::}\alpha \to \alpha\} \vdash \backslash x\ \texttt{->}\ f\ (f\ x) :: \alpha \to \alpha}\,\textsc{Abs}
  }\,\textsc{App}
}{\{\,\} \vdash \backslash f\ \texttt{->}\ \backslash x\ \texttt{->}\ f\ (f\ x) :: (\alpha \to \alpha) \to (\alpha \to \alpha)}\,\textsc{Abs}
$$

A8-3 *Monadische Komposition* In der Vorlesung am 15. Juni wurde das Prinzip demonstriert, wie man mehrere `IO`-Aktionen zu einer einzelnen Aktion verschmelzen kann. Anstatt echtem `IO` wurde der Zustand der Welt durch einen simplen `Integer` beschrieben.

Der Code wurde nun um ein paar einfache Funktionen erweitert, um die Anwendung der Funktionen `nacheinander` und `komposition` zu demonstrieren.

Lösen Sie diese Aufgabe mit Papier & Bleistift, also ohne GHCI einfach nach der Lösung zu fragen!

a) Zu welchem Wert wertet der Ausdruck `demo` aus? Warum und wie?

LÖSUNGSVORSCHLAG:

Zu der Zahl 6.

Der Ausdruck `demo` wendet `demoSequenz` auf eine Welt mit Zustand 0 an und liefert uns danach den Zustand der veränderten Welt zurück.

`demoSequenz` wendet zuerst drei mal hintereinander die Aktion `tick` an. Bei jeder Anwendung der Aktion `tick` wird der Zustand der Welt um eins erhöht, er beträgt also nach den drei Aktionen genau 3. Danach wird noch die Aktion `komposition lese addiere` ausgeführt. `lese` liest den Zustand der Welt aus, also 3. `komposition` liefert das Ergebnis von `lese` an `addiere` weiter, so dass `addiere` noch mal 3 zum Zustand der Welt addiert.

b) Schreiben Sie eine Funktion `tick2 :: IO Int`, welchen den Zustand der Welt bei Aufruf von `tick2` zurückgibt und gleichzeitig den nachfolgenden Zustand der Welt mit 3 multipliziert.

LÖSUNGSVORSCHLAG:

Wenn man den Typ `IO Int` expandiert, bekommen wir `Welt -> (Int,Welt)`. Damit sollte alles klar sein. Wir brauchen lediglich noch die Konversion von `Integer` nach `Int`, welche von der Funktion `fromIntegral` durchgeführt wird, siehe Folie 05-33.

```
tick2 w = (fromIntegral w,w*3)
```

```
import Prelude hiding (IO)

type Welt = Integer
type IO a = Welt -> (a,Welt)

nacheinander :: IO a -> IO b -> IO b
nacheinander f = komposition f . const

komposition  :: IO a -> (a -> IO b) -> IO b
```

```
komposition f g w1 = let (x,w2) = f w1 in g x w2 -- x::a, w2::Welt

-- Ab hier NEUER CODE:
demo = snd $ demoSequenz makeWorld

demoSequenz :: IO ()
demoSequenz = nacheinander tick $
              nacheinander tick $
              nacheinander tick $
              komposition  lese addiere

makeWorld :: Welt
makeWorld = 0

lese :: IO Int
lese w = (fromIntegral w,w)

tick :: IO ()
tick w = ((), succ w)

addiere :: Int -> IO ()
addiere x w = ((),w+(fromIntegral x))
```

H8-1 ***Typherleitung II*** (4 Punkte; Abgabeformat: Text oder 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.

LÖSUNGSVORSCHLAG:

Zur Abkürzung der Notation definieren wir $\Delta = \{f{::}(\alpha \to \alpha) \to \beta \to \beta\}$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Delta, x{::}\alpha \to \alpha \vdash f :: (\alpha \to \alpha) \to \beta \to \beta}\ \textsc{Var} \qquad \dfrac{}{\Delta, x{::}\alpha \to \alpha \vdash x :: \alpha \to \alpha}\ \textsc{Var}}{\Delta, x{::}\alpha \to \alpha \vdash f\ x :: \beta \to \beta}\ \textsc{App}}{\Delta \vdash \backslash x \texttt{ -> } f\ x :: (\alpha \to \alpha) \to \beta \to \beta}\ \textsc{Abs} \qquad \dfrac{\dfrac{}{\Delta, y{::}\alpha \vdash y :: \alpha}\ \textsc{Var}}{\Delta \vdash \backslash y \texttt{ -> } y :: \alpha \to \alpha}\ \textsc{Abs}}{\Delta \vdash (\backslash x \texttt{ -> } f\ x)\ (\backslash y \texttt{ -> } y) :: \beta \to \beta}\ \textsc{App}$$

H8-2 ***Unifikation II*** (4 Punkte; Abgabeformat: Text oder PDF)

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

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

LÖSUNGSVORSCHLAG:

$$\begin{aligned}
&\mathsf{unify}\{\texttt{Int} \to \alpha = \beta \to \texttt{Bool} \to \beta, \gamma \to \texttt{Int} = \alpha\} \\
&\quad = \mathsf{unify}\{\texttt{Int} = \beta, \alpha = \texttt{Bool} \to \beta, \gamma \to \texttt{Int} = \alpha\} \\
&\quad = \mathsf{unify}\{\beta = \texttt{Int}, \alpha = \texttt{Bool} \to \beta, \gamma \to \texttt{Int} = \alpha\} \\
&\quad = [\texttt{Int}/\beta]\mathsf{unify}\{\alpha = \texttt{Bool} \to \texttt{Int}, \gamma \to \texttt{Int} = \alpha\} \\
&\quad = [\texttt{Int}/\beta][\texttt{Bool} \to \texttt{Int}/\alpha]\mathsf{unify}\{\gamma \to \texttt{Int} = \texttt{Bool} \to \texttt{Int}\} \\
&\quad = [\texttt{Int}/\beta][\texttt{Bool} \to \texttt{Int}/\alpha]\mathsf{unify}\{\gamma = \texttt{Bool}, \texttt{Int} = \texttt{Int}\} \\
&\quad = [\texttt{Int}/\beta][\texttt{Bool} \to \texttt{Int}/\alpha][\texttt{Bool}/\gamma]\mathsf{unify}\{\texttt{Int} = \texttt{Int}\} \\
&\quad = [\texttt{Int}/\beta][\texttt{Bool} \to \texttt{Int}/\alpha][\texttt{Bool}/\gamma]\mathsf{unify}\{\,\} \\
&\quad = [\texttt{Int}/\beta][\texttt{Bool} \to \texttt{Int}/\alpha][\texttt{Bool}/\gamma]\mathsf{id} = [\texttt{Int}/\beta, \texttt{Bool} \to \texttt{Int}/\alpha, \texttt{Bool}/\gamma]
\end{aligned}$$

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.

LÖSUNGSVORSCHLAG:
Hier kann man alles Mögliche angeben, was eine der folgenden Gleichungen verletzt:

$$\{\alpha = \delta = \eta, \beta = \gamma = \theta, \}$$

Zum Beispiel: $[\texttt{Bool}/\alpha, \texttt{Double}/\beta, \texttt{Bool}/\gamma, \texttt{Double}/\delta, \texttt{Int}/\eta, \texttt{Int}/\theta]$

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.

LÖSUNGSVORSCHLAG:

Die Typgleichungen der letzten beiden Teilaufgaben sind prinzipiell gleich, was die Unifikation betrifft. Ein Unifikator muss also wieder mindestens folgende Gleichungen lösen:

$$\{\alpha = \delta = \eta, \beta = \gamma = \theta, \}$$

Damit wir einen Unifikator erhalten, welche nicht der allgemeinste ist, müssen wir mehr substituieren als notwendig, also z.B. mindestens einmal einen konkreten Typ einsetzen:

$$[\texttt{Int}/\alpha, \texttt{Int}/\delta, \beta/\gamma, \texttt{Int}/\eta, \beta/\theta]$$

oder aber wir machen alle Typvariablen gleich:

$$[\alpha/\beta, \alpha/\delta, \alpha/\gamma, \alpha/\eta, \alpha/\theta]$$

oder eine Kombination von beiden Möglichkeiten:

$$[\texttt{Bool}/\alpha, \texttt{Bool}/\beta, \texttt{Bool}/\delta, \texttt{Bool}/\gamma, \texttt{Bool}/\eta, \texttt{Bool}/\theta]$$

Abgabe: Lösungen zu den Hausaufgaben können bis Dienstag, den 23.06.2015, 11:00 Uhr mit UniworX abgegeben werden.

Aufgrund des Klausurbonus müssen die Hausaufgaben von Ihnen alleine gelöst werden. Abschreiben bei den Hausaufgaben gilt als Betrug und kann zum Ausschluss von der Klausur zur Vorlesung führen.