The University of Birmingham
School of Computer Science

Solutions for Exercise Sheet 5

Exercise 1: Type checking and typing derivations

Type check the following terms in the applied lambda calculus, and derive their types. In addition, write their typing derivations (either in tree form or linear form).

a. $\lambda x\colon \mathbf{int}.\, \lambda y\colon \mathbf{bool}.\, x.$

Using the method of annotating subterms with their types, we obtain the following. We abbreviate **int** to i and **bool** to b for brevity.

$$(\lambda x\colon i.\, (\lambda y\colon b.\, x^i)^{b\to i})^{i\to b\to i}$$

The formal derivation in the linear form can be written as follows:

1.	$x\colon i,\, y\colon b \vdash x : i$	using *Var*
2.	$x\colon i \vdash \lambda y\colon b.\, x : b \to i$	from 1, using →*Intro*
3.	$\vdash \lambda x\colon i.\, \lambda y\colon b.\, x : i \to (b \to i)$	from 2, using →*Intro*

b. $\lambda p\colon \mathbf{int} \to \mathbf{bool}.\, \lambda f\colon \mathbf{int} \to \mathbf{int}.\, \lambda x\colon \mathbf{int}.\, p\ (f\ x)$

The method of annotating subterms gives:

$$(\lambda p\colon i \to b.\, (\lambda f\colon i \to i.\, (\lambda x\colon i.\, (p^{i\to b}\ (f^{i\to i}\ x^i)^i)^b)^{i\to b})^{(i\to i)\to i\to b})^{(i\to b)\to(i\to i)\to i\to b}$$

The formal derivation is as follows:

1.	$p\colon i \to b,\ f\colon i \to i,\ x\colon i \vdash p : i \to b$	using *Var*
2.	$p\colon i \to b,\ f\colon i \to i,\ x\colon i \vdash f : i \to i$	using *Var*
3.	$p\colon i \to b,\ f\colon i \to i,\ x\colon i \vdash x : i$	using *Var*
4.	$p\colon i \to b,\ f\colon i \to i,\ x\colon i \vdash fx : i$	from 2, 3 using →*Elim*
5.	$p\colon i \to b,\ f\colon i \to i,\ x\colon i \vdash p\,(fx) : b$	from 1, 4 using →*Elim*
6.	$p\colon i \to b,\ f\colon i \to i \vdash \lambda x\colon i.\, p\,(fx) : i \to b$	from 5, using →*Intro*
7.	$p\colon i \to b \vdash \lambda f\colon i \to i.\, \lambda x\colon i.\, p\,(fx) : (i \to i) \to (i \to b)$	from 6, using →*Intro*
8.	$\vdash \lambda p\colon i \to b.\, \lambda f\colon i \to i.\, \lambda x\colon i.\, p\,(fx) : (i \to b) \to (i \to i) \to (i \to b)$	from 7, using →*Intro*

Exercise 2: More type checking

Type check the following term and derive its type:

$$\boldsymbol{Y}\ (\lambda f : \boldsymbol{int} \to \boldsymbol{bool}.\, \lambda n : \boldsymbol{int}.\, \mathit{if}_{\boldsymbol{bool}}\ (n = 0)\ \mathit{true}\ (f\ (n - 1)))$$

Note that $(n = 0)$ and $(n - 1)$ denote syntactic sugar for the function application terms $(=\ n\ 0)$ and $(-\ n\ 1)$ respectively. Here is the term with all its subterms annotated:

$$(\mathbf{Y}\ (\lambda f\colon i \to b.\, (\lambda n\colon i.\, (\mathrm{if}_b^{b\to b\to b\to b}\ (=^{i\to i\to b}\ n^i\ 0^i)^b\ \mathrm{true}^b\ (f^{i\to b}\ (-^{i\to i\to i}\ n^i\ 1^i)^i)^b)^b)^{i\to b})^{(i\to b)\to i\to b})^{i\to b}$$

So, the overall type of the term is **int** → **bool**.

Exercise 3: Type rules

*Formulate type rules for the constructions **let** $x = M$ **in** N and **letrec** $x = M$ **in** N.*

The term **let** $x = M$ **in** N is equivalent to $(\lambda x.\, N)\ M$ in the untyped lambda calculus. However, in the typed lambda calculus, we also need to declare the type of x in $\lambda x.$. Since the function term is applied to M, the declared type of x should be the same as the type of M. Suppose the type of M is T. Then we can produce the type derivation:

$$\dfrac{\dfrac{\Gamma,\, x{:}T \vdash N : T'}{\Gamma \vdash \lambda x{:}T.\, N : T \to T'} \qquad \Gamma \vdash M : T}{\Gamma \vdash (\lambda x{:}T.\, N)\; M : T'}$$

So, we can postulate the typing rule:

$$\dfrac{\Gamma \vdash M : T \quad \Gamma,\, x : T \vdash N : T'}{\Gamma \vdash \textbf{let}\; x = M \;\textbf{in}\; N : T'}$$

Similarly, **letrec** $x = M$ **in** N is equivalent to $(\lambda x.\, N)\,(\mathbf{Y}\,\lambda x.\, M)$ in the untyped lambda calculus. We can proceed as above, by assuming a type T for M.

$$\dfrac{\dfrac{\Gamma,\, x{:}T \vdash N : T'}{\Gamma \vdash \lambda x{:}T.\, N : T \to T'} \qquad \dfrac{\dfrac{\Gamma,\, x{:}T \vdash M : T}{\Gamma \vdash \lambda x{:}T.\, M : T \to T}}{\Gamma \vdash \mathbf{Y}\; \lambda x{:}T.\, M : T}}{\Gamma \vdash (\lambda x{:}T.\, N)\; (\mathbf{Y}\; \lambda x{:}T.\, M) : T'}$$

So, we can postulate the typing rule:

$$\dfrac{\Gamma,\, x : T \vdash M : T \quad \Gamma,\, x : T \vdash N : T'}{\Gamma \vdash \textbf{letrec}\; x = M \;\textbf{in}\; N : T'}$$

Note: Unfortunately, this typing rule is not entirely satisfactory. To type check **letrec** $x = M$ **in** N, we need to type check M first. Since this involves assuming a type for x, which must be the *same* as the type of M, a simple type checker would be unable to use this rule for type checking. ML-like type inference is necessary for using this rule as the basis for type checking.

A better solution for the simply typed lambda calculus would be to use an explicitly typed form of the **letrec** construct with a type declaration for x, e.g., **letrec** $x{:}T = M$ **in** N. The corresponding typing rule for this construct:

$$\dfrac{\Gamma,\, x : T \vdash M : T \quad \Gamma,\, x : T \vdash N : T'}{\Gamma \vdash \textbf{letrec}\; x{:}T = M \;\textbf{in}\; N : T'}$$

does not involve any type inference.

Exercise 4: Type inference

Calculate the principal types for the following terms:

a. $\lambda x.\, \lambda y.\, y\; x$.

The principal type is $t \to (t \to u) \to u$.

Assume types $x{:}\, t_1$ and $y{:}\, t_2$. Then the application $y\; x$ gives the constraints:

$$\begin{array}{rcl} t_2 & = & t_3 \to t_4 \\ t_3 & = & t_1 \end{array}$$

Solving the constraints gives the type $t_1 \to (t_1 \to t_4) \to t_4$. We rename t_1 to t and t_4 to u to obtain a more readable version of the type.

b. $\lambda x.\, \lambda y.\, \lambda z.\, (x\; z)\; (y\; z)$.

The principal type is $(t \to u \to v) \to (t \to u) \to t \to v$.

Assume types $x{:}\, t_1$, $y{:}\, t_2$ and $z{:}\, t_3$. Then the function applications in the body give the following constraints:

$$\begin{array}{rcl} t_1 & = & t_4 \to t_5 \\ t_4 & = & t_3 \\ t_5 & = & t_6 \to t_7 \\ t_2 & = & t_8 \to t_9 \\ t_8 & = & t_3 \\ t_6 & = & t_9 \end{array}$$

Solving the constraints gives the type $(t_3 \to t_9 \to t_7) \to (t_3 \to t_9) \to t_3 \to t_7$. By renaming the type variables for readability we obtain the above type.

c. $\mathbf{Y}\ (\lambda f.\, \lambda x.\, \text{if}\ (\text{null}\ x)\ 0\ (1 + f(\text{cdr}\ x)))$

The principal type is $\textbf{list}\, t \to \textbf{int}$.

Assume the types $f\colon t_1$ and $x\colon t_2$ for the bound variables. Then the body of the function term gives the constraints:

$$\begin{array}{rcl} t_2 & = & \textbf{list}\, t_3 \\ t_1 & = & t_4 \to t_5 \\ t_4 & = & \textbf{list}\, t_3 \\ t_5 & = & \textbf{int} \end{array}$$

The type of the function term is now $(\textbf{list}\, t_3 \to \textbf{int}) \to \textbf{list}\, t_3 \to \textbf{int}$. The application of the **Y** combinator to this term, therefore, has the type $\textbf{list}\, t_3 \to \textbf{int}$. Finally, renaming the type variables for readability we obtain the above type.