Programming Paradigms 2022
Session 2: Types and type classes

Problems for solving and discussing

Hans Hüttel

19 September 2022

1. ***(Everyone at the table together)*** What is the type of the function

```
twice f x = f (f (x))
```

   ?? Explain your answer. Then (and only then) check your answer using the Haskell interpreter. Is the function polymorphic? If it yes, tell us if this is parametric polymorphism or overloading (ad hoc polymorphism). If it is not, tell us why.

2. ***(Solve in pairs)*** What is the type of the function

```
dingo (x,y) = [x,y]
```

   ?? Explain your answer. Then (and only then) check your answer using the Haskell interpreter. Is the function polymorphic? If it yes, tell us if this is parametric polymorphism or overloading (ad hoc polymorphism). If it is not, tell us why.

3. ***(Solve in pairs)*** A famous influencer on Instagram defined a Haskell function bighead that can tell us how many elements in a list xs are greater than (>) the head of xs. As an example of the behaviour of the function instance, the result of bighead [7,4,5,8,9] will be 2. Another influencer was asked what the type of the bighead function is and answered that the type is

```
[Num] -> Num
```

   Why is this not correct? What is the type of bighead?

4. ***(Everyone at the table together)*** Why are function types not allowed to be members of the type class Eq? *Hint:* Many of you have seen something called EQ_{TM} in courses you followed in a past life.

More problems to solve at your own pace

a. Here is a function.

```
mango x y z = x * y + z - 42
```

   What is the type of mango 14? Explain your answer. Then (and only then) check your answer using the Haskell interpreter.

b. Write down a definition of a function bingo that has the following type; it is not important what the definition does as long as it is type correct.

```
bingo :: a -> a
```

   Is bingo polymorphic? If it yes, tell us if this is parametric polymorphism or overloading (ad hoc polymorphism). If it is not, tell us why.

c. Suppose you want to define a function thesame that takes a list of pairs xs and gives us the list of pairs whose first and second component are the same.

   For example, we want the function application

   ```
   thesame [(1,2),(4,4),(6,7),(17,17)]
   ```

   to return the value

   ```
   [(4,4),(17,17)]
   ```

   What should the type of thesame be?

d. Here is a Haskell expression.

   ```
   [ (+), (*), (+), (-) ]
   ```

   What does it contain and what is the type of the expression? Find the answer without asking the Haskell interpreter. Explain why your answer is correct. Then (and only then!) ask the Haskell interpreter what the type is.

   What can you say about the type of

   ```
   [ (+), (*), (+), (-), (++) ]
   ```

   ?

e. Here is a term in the λ-calculus:

   $$(\lambda x.xx)(\lambda x.xx)$$

   Are the bound variables in the term distinct? If they are not, rename them such that they are. Once you have found the answer to this, then find a reduction step that the term can take. To do this, use the reduction rules of the note.

f. Suppose you want to define a function map that takes a function f and a list xs and returns the list where f has been applied to every element in xs.

   For example, suppose double is the function defined by

   ```
   double n = 2 * n
   ```

   Then the function application

   ```
   map double [1,2,3,4]
   ```

   should return

   ```
   [2,4,6,8]
   ```

   What should the type of map be?

g. Find a Haskell expression whose type is

   `(Ord a1, Eq a2) => a2 -> a2 -> (a1, a1) -> a1`

h. Here is the definition of a Haskell function.

   `madras (f,x,y) = f (f x x) y`

   Give a curried version of madras that has type `(t -> t -> t) -> t -> t -> t`,