Programmingsparadigmer 2024
Session 3 : Types and classes

Hans Hüttel

1 October 2024

Our plan for today

1. The learning goals
2. Presentations of the preparation problems
3. Problem no. 1
4. Discussion
5. Break
6. Problem no. 2
7. Problem no. 3
8. Problem no. 4
9. If time allows: More problems at your own pace.
10. We evaluate today's session – please stay until the end!

Learning goals

- To be able to explain and use the central notions of types in Haskell – basic types, list types, function types and tuple types
- To be able to explain and use curried and uncurried functions and how these notions relate to higher-order functions
- To be able to explain and use the notion of type classes
- To be able to explain the notions of polymorphism and overloading (parametric and ad hoc-polymorphism), the difference between them and the relationship to type classes
- To be able to explain and use the syntax and reduction semantics of the pure lambda-calculus
- To be able to explain why renaming of bound variables is a necessary notion

Preparation problem – Types

Write down Haskell definitions of quango and tango that have the following types; it is not important what the definitions do as long as they are type correct.

```
quango :: a -> [a]
tango :: Num p1 => (a, b) -> p2 -> p1
```

Are quango and tango polymorphic? If so, tell us if for each of them if this involves parametric polymorphism or overloading (ad hoc polymorphism) or maybe both – and how. If not, tell us why.

Preparation problem – Reductions in the λ-calculus

Find a terminating reduction sequence of the λ-expression

$$(\lambda x.xy)(\lambda z.(\lambda u.uu))$$

To do this, use the reduction rules of the note.

Problem 1 – twice

What is the type of the function

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

?? Explain your answer and *how you found it*. 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.

What about the function

```
twicetwo (f,x) = f (f (x))
```

?

Discussion – Help the influencer

A famous influencer on TikTok 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?

Break

Problem 2 – The λ-calculus

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.

Problem 3 – dingo

What is the type of the function

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

??

Explain your answer and *how you found it*. 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.

Discussion: But why . . . ?

A politician wrote the following piece of code:

```
plip :: Integer -> (Integer,Integer)

plip x = (x,x)
```

and said that plip is a function that takes an integer and gives us a pair of integers as its value.

The politician's friend, a famous television personality, exclaimed: "Sorry, but that is not correct. The plip function is polymorphic! Just look at its definition."

Who is right here?

Problem 4 – Function types and equality types

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.

Working at your own pace (pair programming)

There are further problems in the problem set, that you can work on during the rest of the session. See if you can solve one or more of them.

Evaluation

- What did you find difficult?
- What surprised you?
- What went well?
- Is there a particular problem that we should follow up on with a short video?