Programming Paradigms 2025
Session 7: Declaring types and type classes

Problems for solving and discussing

Hans Hüttel

22 October 2025

Problems that we will definitely talk about

1. ***(10 minutes)***

   Define a Haskell datatype Aexp for arithmetic expressions with addition, multiplication, numerals and variables. The formation rules are

   $$E ::= n \mid x \mid E_1 + E_2 \mid E_1 \cdot E_2$$

   Assume that variables x are strings and that numerals n are integers.

2. ***(20 minutes)***

   Use your Haskell datatype from the previous problem to define a function eval that can, when given a term of type Aexp and an assignment ass of variables to numbers compute the value of the expression.

   As an example, if we have the assignment $[x \mapsto 3, y \mapsto 4]$, eval should tell us that the value of $2 \cdot x + y$ is 10.

   Hint: An assignment is a function. How can you represent it?

3. ***(20 minutes)***

   The authors of a new encyclopedia about values of type a represent their information using a hierarchical structure where every entry in the encyclopedia is tagged with a key that is a string and a value of type a. Entries can have subentries.

   Figure 1 shows two encyclopedias.

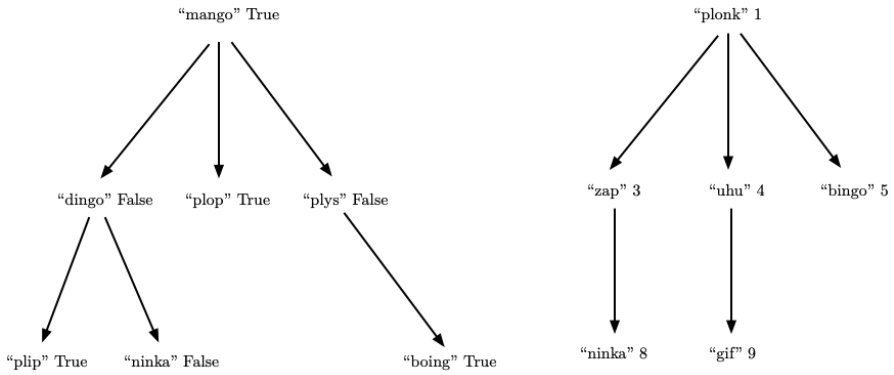

Figure 1: Two encyclopedias t_1 (left) and t_2 (right)

The authors of the encyclopedia asked a famous influencer to define a type Encyclopedia for encyclopedias. The influencer wrote

```
data Encyclopedia a = Leaf String a |Node1 String a (Encyclopedia
a) | Node2  String a (Encyclopedia a) ( Encyclopedia a)| Node3
String a (Encyclopedia a) ( Encyclopedia a) (Encyclopedia a)
deriving Show
```

but complained about the limitations of Haskell: *"I see from the examples that any entry can have no more than three sub-entries. Nothing in Haskell allows you to write that there are arbitrarily many children of the same type, so I am forced to write a very clumsy definition. Haskell is pathetic!"*

- Provide a better definition. Find out what is wrong and come up with a better solution. It is a good idea to read the problem text very carefully and find out what it says and does not say. Once you have criticized the existing solution, it is also a good idea *not to try to repair* the existing attempt but to start over.
- Show how you represent t_1 and t_2 in Figure 1 as values t1 and t2 of your type.

4. ***(20 minutes)***

   An encyclopedia is *layered* if it holds that all values at the same level of the encyclopedia are larger than the values in the levels above. As an example, t_2 in Figure 1 is layered, since 8 and 9 at level 3 are greater than the values 3, 4 and 5 at level 2 – which are greater than the value 1 at level 1.

   Define a function layered that can tell us if an encyclopedia is layered.

5. ***(20 minutes)***

   From linear algebra we know that a vector space with inner product is one for which the operations of vector sum and dot product are defined. Given two vectors v_1 and v_2, the sum $v_1 + v_2$ is again a vector, and the inner product $v_1 \cdot v_2$ is a number. *Please note:* In linear algebra there is much more to the definition of inner product spaces than these two requirements, but in this problem, please ignore that. Also assume that the inner product is a number of type Int.

   - Define a typeclass InVector whose instances are types that can be seen as inner product spaces, where vector sum is called &&& and inner product is called ***. *Hint:* Which section in the text for today do you need here?
   - Find out how to declare Bool as an instance of InVector. *Hint:* You have to find a definition of vector sum and inner product for truth values.

More problems to solve at your own pace

In your solutions, remember the learning goals of this session!

a) We say that a binary tree is *balanced* if the number of leaves in every left and right subtree differ by at most one with leaves themselves being trivially balanced. Define a function balanced that will tell us if a binary tree is balanced or not. *Hint:* It is a good idea to also define a function that finds the number of leaves of a tree.

b) This problem refers to Section 8.6 in the book and the types and functions defined there.

   Two Boolean propositions p and q are *equivalent* if they are true for the same substitutions, that is, for every substitution s we have that p is true under s if and only if q is true under s. Define a function equiv where

```
equiv :: Prop -> Prop -> Bool
```

   and such that equiv p q returns True if p and q are equivalent and False otherwise. *Hint:* What does "if and only if" mean in logic?

c) On page 106 there is a definition of the Expr datatype for expressions. Define a higher-order function

```
foldexp :: (Int -> a) -> (a -> a -> a) -> Expr -> a
```

   such that foldexp f g replaces each Val constructor in an expression by the function f, and each Add constructor by the function g.

   Then use foldexp f g (with appropriate choices for f and g) to define a function eval :: Expr -> Int that evaluates an expression to an integer value.

d) Complete the following instance declaration:

```
instance Eq a => Eq (Maybe a) where
...
```