Programming Paradigms 2024
Session 7 : Declaring types and type classes

Hans Hüttel

29 October 2024

Our plan for today

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

Learning goals

- To understand how type, newtype and data declarations work and how they differ.
- To be able to define functions over recursively defined data types using pattern matching.
- To understand how recursively defined data types can be used to implement the formation rules of a language.
- To understand the notion of term constructors and how they are used.
- To understand the principles of and be able to declare new instances of type classes.
- To understand how the above notions can be applied in the setting of a larger program.

Preparation problem – Unary numerals

Unary numerals consist of a finite sequence of `I`'s followed by a `Z`. The natural number n can be represented as n successive `I`'s and a `Z`, so e.g. 4 is represented in unary notation as `IIIIZ`. The natural number 0 is represented as `Z`.

Define a recursive datatype Unary for unary numerals and use your type definition to define a function unary2int of type unary2int :: Unary -> Integer that finds the natural number represented by a given number. As an example, unary IIIIZ should give us 4

Preparation problem – least

Use the declaration of the type Tree on page 97 to define a function least that finds the least element in a given binary tree.

What should the type of least be?

Problem 1 – Expressions (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.

Problem 2 – Evaluating expressions (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.

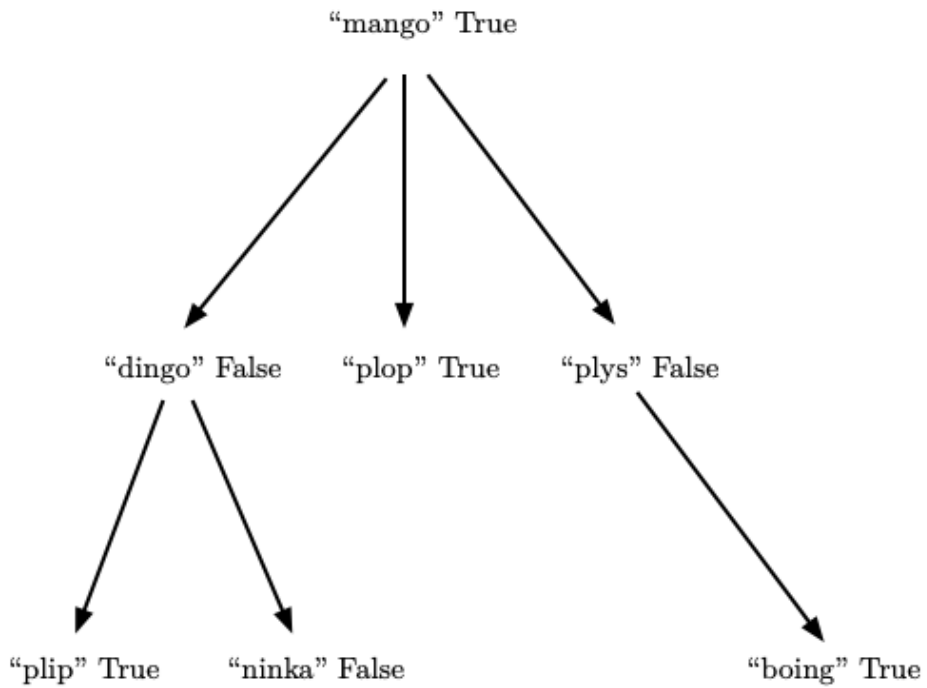
Hint: An assignment is a function. How can you represent it?

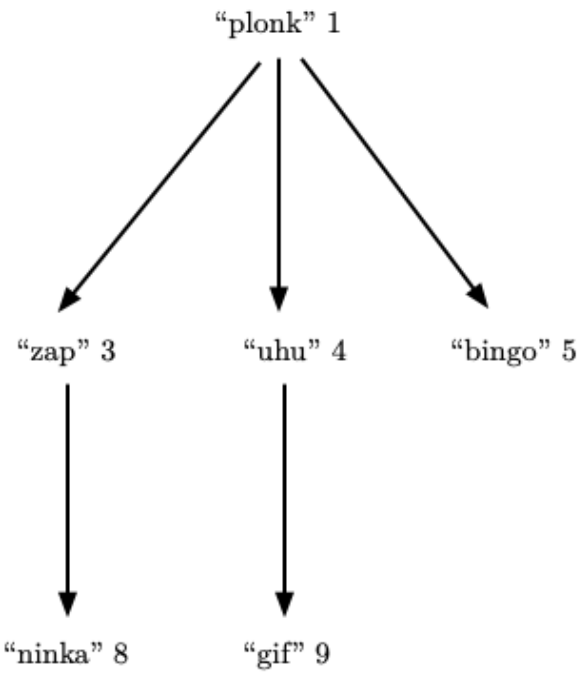
Break

Discussion – An encyclopedia

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.

Here are two encyclopedias (also shown in the problem set).





Discussion – An encyclopedia

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!"*

Problem 3 – A better solution (20 minutes)

- Provide a better definition of the Encyclopedia type. 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 from before as values t1 and t2 of your type.

Problem 4 – Layered encyclopedias (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 from before 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.

Problem 5 – Defining your own type class (20 minutes)

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 this! 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 ***.

Find out how to declare Bool as an instance of InVector.

Evaluation

- What did you find difficult?
- What surprised you?
- What went well?
- What could be improved? How does the setup work this time?
- Is there a particular problem that we should follow up on with a short video?