Q&A Session for Programming Languages Lecture 20

Session number: 1203582659
Date: Tuesday, November 17, 2020
Starting time: 2:18 PM

ANON - 2:29 PM
Q: Is there a quiz? I don't see it on Submitty
Priority: N/A
        Konstantin Kuzmin - 2:30 PM
        A: It will go live at 2:30 pm EST.

ANON - 2:45 PM
Q: Did we ever go over appending to lists in haskell? how would we do that?
Priority: N/A
        Ana L. Milanova - 2:46 PM
        A: We did go over cons.

ANON - 2:50 PM
Q: My group's submissions for Question 3 are showing and error saying the file is not found
Priority: N/A
        Konstantin Kuzmin - 2:56 PM
        A: Thank you for noticing this. We are looking into this issue.
        Konstantin Kuzmin - 2:57 PM
        A: It is safe to ignore this error. Your files are successfully saved.

ANON - 3:05 PM
Q: I was under the impression that Haskell was statically typed or did I just hear you wrong?
Priority: N/A
        Steven Haussmann - 3:10 PM
        A: It is statically typed, yes. Compilation will fail if it can't typechecl.

ANON - 3:17 PM
Q: how does haskell decide when is time to evaluate, given that it's lazy?
Priority: N/A
        Steven Haussmann - 3:20 PM
        A: It evaluates something when it actually needs the value --

meaning that it's not just being passed around, but actually read
        Ana Milanova - ???
        A: Yes, I second what Steven says. At a high level, Haskell will evaluate when the value is needed, e.g., it is a function applied on arguments, or it is an argument to an arithmetic operation, etc. The actual rules are more complex though!

ANON - 3:19 PM
Q: is there any difference between x : [y] and [x]++[y] ?
Priority: N/A
        Steven Haussmann - 3:21 PM
        A: No, barring potential efficiency differences (the latter makes two lists, rather than one)

ANON - 3:23 PM
Q: Soin Scheme cons with vary types will succeed, but Haskell fails because of its Static Type Checking?
Priority: N/A
        Ana L. Milanova - 3:27 PM
        A: Essentially, yes. Haskell will assign a type for the list, [T] which would demand that the elements are all of the same type T.

ANON - 3:24 PM
Q: how is it possible to infer types?
Priority: N/A
        Steven Haussmann - 3:26 PM
        A: It's basically logical induction. For example, if a function takes an argument and applies head to it, that argument has to be a list (of some unknown type)

ANON - 3:25 PM
Q: Will Haskell's type inference always find the most specific possible type of a variable based on the context in which the variable is used?
Priority: N/A
        Steven Haussmann - 3:26 PM
        A: No, it will be the most general type possible.
        Steven Haussmann - 3:27 PM
        A: The type of f l = head l will be [a] -> a, where a is any type
        Ana L. Milanova - 3:27 PM
        A: Yes, what Steven says. The so-called principal type in the Hindley-Milner type system. Haskell's type system is an extension of the well-known Hindley Milner, which entails certain kind of inference, including inference of the most general (i.e., principal)

type.

ANON - 3:29 PM
Q: Will Haskell throw some error or exception if a variable could be inferred to be multiple different types?
Priority: N/A
Steven Haussmann - 3:30 PM
A: Only if that results in an inconsistency. If one place requires [a] and one places requires [Int], that's fine.
Ana Milanova - ???
A: In general yes. But there are cases when Haskell will throw an error if a variable is "polymorphic", i.e., it is one type in one context and another type in another context, and Haskell cannot deal with the polymorphism. Haskell, based on Hindly Milner, admits a limited form of polymorphism known as let-polymorphism.

ANON - 3:31 PM
Q: I'm still not familiar with the Haskell syntax
Priority: N/A

ANON - 3:37 PM
Q: I've noticed that "a":["a"] works but Cons "a" ["a"] does not. Why? What is wrong in the syntax?
Priority: N/A
Steven Haussmann - 3:38 PM
A: Cons is not part of the language. This is something we've come up for an example.
Ana L. Milanova - 3:40 PM
A: Yes, what Steven says. Cons is just one of the data constructors for the List ADT that we defined as an example. It is not a built-in function.

ANON - 3:38 PM
Q: what is a generic function?
Priority: N/A
Steven Haussmann - 3:39 PM
A: A function that can work with multiple types. For example, the head function takes a list of any type and returns something of that type

---

ANON - 3:40 PM
Q: I'm still not familiar with the Haskell syntax, are there any other tips beside practicing?
Priority: N/A
Ana L. Milanova - 3:41 PM

A: The examples from lecture should be sufficient to tackle the homework. Haskell's syntax is very rich and I've tried to focus on a minimal subset to get us started with.

Ana L. Milanova - 3:42 PM

A: You can type those expressions. Keep practicing by typing expressions using built-in functions into the interpreter. Then define your own functions and start calling those functions, etc.

Steven Haussmann - 3:43 PM

A: At the same time, the basics required for writing programs aren't too complex -- I'd do some reading (there are a wealth of great Haskell tutorials out there) and try writing small functions

---

ANON - 3:42 PM
Q: what are the conditions for a type class exactly? Is there a list of artihmetic operations it must satisfy?
Priority: N/A

Steven Haussmann - 3:45 PM

A: It's a set of function signatures that must be implemented. It's like an interface.

ANON - 3:43 PM
Q: What is the => Syntax mean? Is it just saying for all of the occurences of a must be a Num?
Priority: N/A

Ana L. Milanova - 3:44 PM

A: Yes, that is correct. It constrains the instantiations of the type parameter a. It states the constraint that valid type arguments (i.e., instantiations of the type parameter a) must be instances of type class Num

---

ANON - 3:47 PM
Q: So Type Class Instance = Type Class Specific To A Specific Type Like Integer, ...
Priority: N/A

Steven Haussmann - 3:48 PM

A: An instance of a typeclass is a type satisfying the rules for that typeclass.

Steven Haussmann - 3:49 PM

A: integers are an instance of the Ord typeclass, because it supports all of the functions the typeclass requires

---

ANON - 3:49 PM
Q: I still don't really understand what a type class is?
Priority: N/A

Steven Haussmann - 3:51 PM

A: They let us require things of types. A function of type Ord

a => a -> a -> Bool, for example, takes two things of the same type, with the caveat that they satisfy the Ord typeclass, and returns a bool

Steven Haussmann - 3:52 PM

A: A valid implementation of that signature: check a b = a < b

Steven Haussmann - 3:53 PM

A: If you look up the definition of Ord, you'll see it requires implementation of <, >, <=, >=, and so forth

---

ANON - 3:53 PM

Q: I know there's not a great definition, but how are monads used?

Priority: N/A

Ana L. Milanova - 3:54 PM

A: We'll see two common use cases: Maybe and List.

ANON - 3:58 PM

Q: for this slide, what exactly is it saying?

Priority: N/A

Steven Haussmann - 3:59 PM

A: It showed that, without taking advantage of monadic behavior, we had to explicitly check if we had a value or not at each step.

ANON - 3:58 PM

Q: So I'm confused if a Monad is m, or if the Monad is m a. Why does the Class Monad only mention m if it is to be m a?

Priority: N/A

Steven Haussmann - 4:01 PM

A: "Monad m" means "some type m that satisfies the Monad typeclass"

Steven Haussmann - 4:02 PM

A: So m is, indeed, an instance of the Monad typeclass

ANON - 4:00 PM

Q: is monad similar to recursion?

Priority: N/A

Steven Haussmann - 4:01 PM

A: No, they aren't much related. Recursion is a function that invokes itself; monads contain information and allow it to be operated on (but possibly without letting us access that information)

ANON - 4:02 PM

Q: What is the difference between m and m a? Is m a = Monad m that takes the Argument a ?

Priority: N/A

Steven Haussmann - 4:03 PM
A: "m a" means some type m, with a single type argument, a
Steven Haussmann - 4:03 PM
A: Much like how a list takes a type argument to determine what it contains.

ANON - 4:04 PM
Q: If m a has a as the Type Argument, why is it not the case that the class Monad is defined with m and a similar to the Eq a class?
Priority: N/A
Steven Haussmann - 4:06 PM
A: The typeclass asserts that, to be a Monad, you must implement a few functions. Those functions are required to operate on a type that is a Monad, but also have additional types (which aren't also Monads!)

ANON - 4:05 PM
Q: I think Professor Milanova meant to write >>= as opposed to f for that Type Annotation here as the type of f is a -> ma
Priority: N/A
Ana Milanova - ???
A: I will go back and double check the video, I might have made a mistake. I can't remember off the top of my head.

ANON - 4:07 PM
Q: Are we required to use the bind and return operations of the Maybe monad on the homework? I initially implemented the homework using cases on Nothing and Just
Priority: N/A
Steven Haussmann - 4:07 PM
A: No, you are not. You might find it helpful for conciseness.

ANON - 4:08 PM
Q: But I'm still confused why the Definition of the Monad does not say class Monad m a where ... since the Monad would be Generic?
Priority: N/A
Steven Haussmann - 4:11 PM
A: The Monad typeclass doesn't place any constraints on the type we use for a. That can be done later, however: e.g. Maybe contains a specific type

---

ANON - 4:09 PM
Q: what does the list monad do exactly?
Priority: N/A

Steven Haussmann - 4:16 PM
A: It lets us operate on the data in lists with a series of functions. The functions don't care about how long the list is -- they just operate on individual elements (which match the type contained in the list)

ANON - 4:16 PM
Q: What exaclty does . f do? I know she said Function Compositoon but I'm confused about how that works with >>=
Priority: N/A
Steven Haussmann - 4:18 PM
A: It's composing the return function with f. The resulting function applies return to the result of applying f to its argument.
Steven Haussmann - 4:18 PM
A: arithmetic example: ((* 2) (+ 1)) 5 produces 12. It adds 1, then multiplies by 2.

ANON - 4:19 PM
Q: So the Previous Slide would apply F(G()) On All Elements From The [1,2,3]?
Priority: N/A
Steven Haussmann - 4:21 PM
A: It will, for each element in the list, apply the function, then return them -- which produces a one-element list. These lists are then joined together.

ANON - 4:22 PM
Q: Is >>= Right/Left Associative?
Priority: N/A
Ana Milanova - ???
A: Monads come with the so-called Monad laws, which require, among other things, that the bind is associative. It is up to the programmer to ensure associativity though.