CIS 1904: Haskell

Course Staff

Instructor: Me (Cassia Torczon)

- 3rd year PhD student in programming languages
- Office Hours: 2:30-3:30 on Tuesdays, or by appointment
    - Location: GRW 571

TA: Matthew Pattok

- Undergraduate in computer science
- Office Hours: TBD

Course Policies

Homework:

- Roughly weekly, due Wednesdays at 11:59pm
- Please complete individually (do **not** use Copilot or ChatGPT)
  - You may use Hoogle and discuss *high-level* ideas with classmates
- 3 total late days
  - These are atomic; 1h late and 23h late are the same
  - If for whatever reason you find you need more, please email me!
    The sooner I know, the easier it is to arrange accommodations.

Course Policies

Class:

- Attendance is required
  - Please do not come if you’re sick! Just email me, preferably before class.
  - Grades will be roughly 10% participation, 90% homework
- Mostly lecture, with some in-class exercises
- Some classes may have a reading to do beforehand

We will use:

- GitHub: for distributing homework
- Canvas: for final grades
- Ed: for asking questions
- Gradescope: for turning in homework
- Poll Everywhere: for some in-class exercises
- Course website: these policies, general course information

What is this class about?

Haskell

What is this class about?

Haskell as a key example of a programming language that is:

- functional
- pure
- lazy
- statically-typed

What is a functional language?

- Functional programming: a *paradigm* built around **applying** and **composing functions**
- Designed to “look like math”
  - Evolved out of logic
- A few alternatives to functional programming:
  - Imperative programming (common in C)
  - Object-oriented programming (common in Java)

What is a functional language?

- A language built around **application** and **composition** of **functions**
  - Follows the *functional programming paradigm*
- Designed to “look like math”
  - Evolved out of logic
- A few other paradigms:
  - Imperative programming (common in C)
  - Object-oriented programming (common in Java)

OCaml is designed to mix all of these!

What is a pure language?

- No side effects:
  - Printing
  - Reading from memory
  - Anything except evaluating a term down to a simpler term
    - e.g. $2 + 2 + 2 \rightarrow 4 + 2 \rightarrow 6$
- Easy to reason about
- Helps avoid classes of bugs (esp. concurrency bugs)
- We will see later how Haskell manages this
  - (It's monads)

What is a lazy language?

In a lazy language, programs do not get evaluated until their results are needed!

```
foo :: Int → Bool
foo x = True
```

What happens if we call `foo(217`^{81}`)`?

What is a lazy language?

In a lazy language, programs do not get evaluated until their results are needed!

```
foo :: Int → Bool
foo x = True
```

What happens if we call `foo(217^81)`?

- In most languages, we will have to calculate 217^{81} before returning `True`.
- In Haskell, we can notice that we never need to know the value of x to evaluate the function and just return `True` right away!

What is a statically-typed language?

- Static typing: types are known *at compile time*
  - e.g., the programmer annotates terms with their types (as in OCaml)
  - e.g., the compiler is able to infer types for many terms (also as in OCaml)
  - facilitates analysis and optimization during compilation
  - immediate feedback and refactoring support from IDE
  - provides documentation and design support
- Dynamic typing: types are determined at *runtime*
  - e.g., as in Python
  - often quicker to write without annotations
  - sometimes gives the programmer more freedom

Why Haskell?

- It’s an excellent example of a functional, pure, lazy, statically-typed language
- Useful tool for exploring abstractions
- It’s fun :)

Installation!

https://github.com/cassiatorczon/cis1904-spring25/blob/main/week01/homework/instructions.md