CSC324 Notes

Autumn 2025
Stefan Barna

Contents

Haskell is a strongly typed functional programming language
- very good for defining domain-specific languages (DSLs); examples of DSLs include SQL and CSS

What defines a programming language?
- syntax, semantics, underlying fundamental concepts

How are programming languages designed and implemented?

How do we reason about programs?
- what properties does a program have? how do we prove them?

What is being researched wrt programming languages?
- Theoretical: Formal semantics, logics (CH isomorphism), type theory, program analysis, proof systems/assistants,...
- Applied: compiler design and optimizations, domain-specific languages, concurrency, distributed systems

Questions What are algebraic data types? What is type theory?

> Note 1 : Read chapters 1, 2 of TAPL after lecture.

1 Types and Programming Languages (Introduction)

Type systems are a lightweight formal method.

> Definition 2 **type system** : A *type system* is a tractable syntactic method for proving the absence of certain program behaviours by classifying phrases according to the kinds of values they compute.

Typing rules are usually applied to the code's syntax, classifying terms according to the properties of the values that they will compute when executed.

Statically typed programming languages are those for which types are known at compile time. Dynamically typed languages, in contrast, are those for which types are only known at runtime.

We will focus on type systems found in programming languages, while type systems (or *type theory*) more generally refers to a much broader field of study in logic, mathematics, and philosophy.

Types introduce order to a language by defining what kinds of data can interact with what other kinds of data, and how. The stronger the type system, the safer the language.
- language design goes hand-in-hand with type system design

2 Haskell

Haskell is a *purely functional* programming languages.

- There are no side effects.
- Functions are first-class citizens, meaning we can pass functions as inputs, and return them as outputs.
- Haskell is lazy, meaning computations will not happen unless necessary.
- Haskell is strongly typed with a powerful type inference system.

Variables are not mutable in Haskell.