13 - Functional Programming

The Church's thesis states that any problem that can be solved by an algorithm can be solved by a Turing machine and vice versa. It implies that there are fundamental limits to what can be computed by any mechanical or algorithmic means. It also underlies the development of modern digital computers and serves as the theoretical foundation for much of the field of theoretical computer science.

Church also introduced lambda calculus, which is a universal model of computation: any computable function can be expressed and evaluated in the lambda calculus. It is consistent with the Church-Turing thesis, and the thesis can be used to argue that the lambda calculus is a powerful and flexible model of computation that can be used to solve a wide range of problems.

Lambda calculus was inspirational for functional programming, where the key idea is to do everything by composing functions. This allows to minimize/eliminate side effects, which are any modification of state that occurs as a result of evaluating an expression. It would be desirable to avoid them, as they make it difficult to reason about the behaviour of a program, potentially introducing bugs/unexpected behaviour. Functional programming languages achieve this property in different ways, such as using immutable data structures, higher-order functions, and pure functions.

Features

- Functions:
  - 1st class citizens: treated as values that can be passed as arguments, returned as results, and stored in variables;
  - Higher-order: can take other functions as arguments or return functions as results;
  - Pure: no side effects;
- Immutable data;
- Lazy evaluation: expressions are not evaluated until they are needed. This potentially allows the creation of infinite data structures, better performance and shorter code;
- Recursion: take place over iteration;
- Polymorphism, typically universal parametric implicit;
- Garbage collection;

ML family

The ML (Meta Language) family of programming languages is a group of programming languages that share a core set of features:

- Static type checking;
- Type inference;
- Pattern matching;
- Algebraic data types;
- Higher-order functions;
- Module system;
- Garbage collection.

The ML family of languages has been influential in the development of functional programming languages and type systems, and continues to be used in academic and industry settings.