8 Nov 2018.

10 The Lambda Calculus

1. Motivation

① Most programming languages support a mechanism for declaring functions and applying them to arguments. In fact, in functional languages, such as Ocaml, Haskell, Clojure and Scala (to some extent), function declaration and application is the main device of computation (not state access and update).

② The lambda calculus is a simple formal language that lets us study principles behind function declaration and application without being distracted by the complexities of usual real-world programming languages. It forms the basis of many functional languages. Also, it can be used to define a notion of computability.

③ One interesting construct of the lambda calculus is so called lambda abstraction:

$$\lambda x.\, e$$

which denotes a function with formal argument x and body e. Nowadays most mainstream languages (C++, Java, Python, ...) support this construct. The lambda abstraction is particularly useful when we use higher-order functions. For instance, to express

$$\int_0^1 dx \int_x^x dy\ (x+y)^2$$

in a prog. language with the integrate primitive, we can write.

integrate (0, 1, λx. integrate(0, x, λy. (x+y) x (x+y)))

using lambda abstraction. But without it, we should write