

Higher Order Operator Overloading With Function Currying

by Sven Nilsen, 2019

In higher order operator overloading^[1], one has the following:

$$g(f_0, f_1) := \lambda(a: A) = g(f_0(a), f_1(a))$$

$$f_0 : A \rightarrow T$$

$$f_1 : A \rightarrow T$$

$$g : T \times T \rightarrow T$$

Assume the following types instead:

$$f_0 : A \rightarrow B \rightarrow T$$

$$f_1 : A \rightarrow B \rightarrow T$$

$$g : T \times T \rightarrow T$$

Notice that the function f_0 and f_1 takes two arguments by function currying^[1].

Starting with two arguments, one can work backwards using higher order operator overloading^[1]:

$$g(f_0(a)(b), f_1(a)(b)) : T$$

$$g(f_0(a), f_1(a)) : B \rightarrow T$$

$$g(f_0, f_1) : A \rightarrow B \rightarrow T$$

This proves that higher order operator overloading can be used with function currying.

References:

- [1] “Higher Order Operator Overloading”
Sven Nilsen, 2018
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/higher-order-operator-overloading.pdf

- [2] “Currying”
Wikipedia
<https://en.wikipedia.org/wiki/Currying>