Higher Order Operator Overloading With Function Currying

by Sven Nilsen, 2019

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

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

$$f_0 : A \to T$$
$$f_1 : A \to T$$
$$g : T \times T \to T$$

Assume the following types instead:

$$f_0 : A \to B \to T$$
$$f_1 : A \to B \to T$$
$$g : T \times T \to T$$

Notice that the function `$f_0$` and `$f_1$` takes two arguments by function currying.

Starting with two arguments, one can work backwards using higher order operator overloading:

$$g(f_0(a)(b), f_1(a)(b)) : T$$
$$g(f_0(a), f_1(a)) : B \to T$$
$$g(f_0, f_1) : A \to B \to T$$

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