

Higher Order Operator Overloading and Partial Evaluation

by Sven Nilsen, 2019

Higher Order Operator Overloading states that when you have functions of the same input, you can overload any operator on the return values of the functions:

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

Self Reference under Higher Order Operator Overloading means constructing functions like these:

$$x := \lambda(x : \text{real}, y : \text{real}) = x$$

$$y := \lambda(x : \text{real}, y : \text{real}) = y$$

Partial Evaluation means calling some function `f` and interpreting it using Higher Order Operator Overloading, with one or more self-referential function:

$$f(x : \text{real}, y : \text{real}) = x^2 + y^2 \leq 1$$

$$f(x, 0) \leq \Rightarrow \lambda(x : \text{real}, y : \text{real}) = x^2 \leq 1 \quad \text{Partially evaluated function where `y = 0`}$$

$$f(0, y) \leq \Rightarrow \lambda(x : \text{real}, y : \text{real}) = y^2 \leq 1 \quad \text{Partially evaluated function where `x = 0`}$$

To reduce the number of arguments, one can use the `id` function:

$$f(x : \text{real}, y : \text{real}) = x^2 + 3 \cdot y$$

$$f(\text{id}, 0) \leq \Rightarrow \lambda(x : \text{real}) = x^2$$

$$f(0, \text{id}) \leq \Rightarrow \lambda(y : \text{real}) = 3 \cdot y$$

If there are more than one argument, one must use self-referential functions of the new context:

$$f(x : \text{real}, y : \text{real}, z : \text{real}) = x^3 + y^2 + z$$

$$x := \lambda(x : \text{real}, z : \text{real}) = x$$

$$z := \lambda(x : \text{real}, z : \text{real}) = z$$

$$f(x, 0, z) \leq \Rightarrow \lambda(x : \text{real}, z : \text{real}) = x^3 + z$$

One can also extend the context, e.g. to add infinitesimal changes:

$$x := \lambda(x : \text{real}, dx : \text{real}, y : \text{real}, dy : \text{real}, z : \text{real}, dz : \text{real}) = x + dx$$

Partial Evaluation is different from Function Currying, although it is logically equivalent:

- Partial Evaluation reduces a program into basis function-calls
- All constant expressions are reduced

For example:

$$f(x) = x + (1 + 4)$$

When passing `x` as a self-referential function, `f` becomes:

$$f(x) = x + 5$$

Notice that this is a logically equivalent function, but it might run faster.

So, naturally, Higher Order Operator Overloading might be used to:

- Analyze the inner workings of programming languages
- Change performance characteristics of functions