Higher Order Operator Overloading Lifting

by Sven Nilsen, 2019

Assume the following equation using Higher Order Operator Overloading[1] (HOOO):

```
f = a + g

f : X → real
g : X → real
a : real
```

Expanded to full closure/lambda[2] form:

```
\(x : X) = f(x) == a + g(x)
```

Higher order operator overloading lifting is done by replace `a` with a function `h`:

```
h := \(x : X) = a
```

Such that the expanded version can be written in an equivalent form:

```
\(x : X) = f(x) == h(x) + g(x)
```

Since this is an equation with only functions, it can be written in pure HOOO form:

```
f = h + g
```

Now, by swapping all occurences of `a` with `h`, one can think of `a` being transformed into `a'`:

```
f = a' + g

a' := \(x : X) = a
```

Or, simply:

```
f = a + g
```

This makes any mixture of function and scalar types possible, as an extension to pure HOOO syntax.

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] “Lambda Notation”
Sven Nilsen, 2018
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/lambda-notation.pdf