Higher Order Operator Overloading and Existential Path Equations

by Sven Nilsen, 2019

In this paper I introduce a notation for Existential Path Equations.

In dependent type systems[1], the same syntax might be used on the type level as for values. I previously introduced a notation[2] for expressing explicitly when Higher Order Operator Overloading[3] (HOOO) is used for some parameter. By applying this new notation on value level instead of type level, one gets an interpretation that corresponds to defining the higher order function by its existential path.

For example, in propositional logic, a proof is a function that returns `true` for all inputs:

f <=> \true		`f` is a proof

The meaning of this is that the existential path `∃f` returns `true` for `true` and `false` otherwise:

∃f <=> (= true)

Although the number of arguments and their types of the function `f` are unknown, the identity of this higher order function is still defined by the existential path. It is known that the function `f` returns `true` for all inputs, no matter what they are, so if you give me the input type, I can always construct the full function. One can also think about it as the `<=>` operator making an inference at HOOO level when one of the sides uses `\`. This is an Existential Path Equation.

Bi-directionality of the function identity only holds when the existential path returns `true` for some unique output value. One can use a directional arrow when there is a non-unique solution:

f => \(> 2)		The existential path is inferred but the identity of the function is unknown

∃f <=> (> 2)		Using the existential path explicitly gives less information about identity

If the function returns a unique output value for all inputs, then for any non-empty domain constraint the same existential path equation holds (it is a uniform set property[4] except for the empty set):

f <=> \true
f{(> 8)} <=> \true

A sub-type of a function by using the universal existential path[5] might be translated into an equation:

f : [∃] (= true)	f : [∃] (> 2)
f <=> \true	f => \(> 2)

References:

[1] "Dependent type"
Wikipedia
https://en.wikipedia.org/wiki/Dependent_type

[2] "Higher Order Operator Overloading and Notation for Parameters"
Sven Nilsen, 2019
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/higher-order-operator-overloading-and-notation-for-parameters.pdf

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

[4] "Uniform Properties of Sets"
Sven Nilsen, 2018
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/uniform-properties-of-sets.pdf

[5] "Universal Existential Paths"
Sven Nilsen, 2017
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/universal-existential-paths.pdf