Predicate Interpretation of Adjoint Paths

by Sven Nilsen, 2020

When encoding functions as predicates[1], it is common to use the following technique:

$$p(A, B) : bool$$

Where `p` is the predicate, `A` is input and `B` is output.

A left adjoint[2] g_0 and a right adjoint g_1 can be thought of as a relationship:

$$\forall\ A, B\ \{\ p(g_0(A), B) = p(A, g_1(B))\ \}$$

This would mean that `p` as a function:

$$p(g_0(A)) = B \qquad p(A) = g_1(B)$$

In that sense, one can understand that g_0 and g_1 has an input/output relationship of `p`.

The difference is that an adjoint path[3] generalizes over any type with equality:

$$p(A, B) : T$$

$$eq : T \times T \to bool$$

The predicate interpretation is not meant to be taken literally, but to make it easier to understand why adjoint paths encodes relations that are similar to the input/output relationship of functions.

In some sense, using the same encoding, one can say that:

$$p(A, g_1(B))$$

Defines what `p` returns for `A`, given that `A = B`.

Similarly, one can say that:

$$p(g_0(A), B)$$

Defines what `p` must have as input for the output `B`, given that `A = B`.

This interpretation depends entirely on the choice of using the second argument as output.
By choosing the first argument as output, the interpretation becomes the reverse.

However, no matter which encoding one chooses, the intuition of non-self-adjoint paths overlaps the intuition of input/output of functions such that one side is thought of as the “input” and the other side as the “output”. This might not always be the case, but it could be a helpful interpretation sometimes.

References:

[1] “First-order logic”
Wikipedia
https://en.wikipedia.org/wiki/First-order_logic

[2] “Adjoint functors”
Wikipedia
https://en.wikipedia.org/wiki/Adjoint_functors

[3] “Adjoint Paths”
Adam Nemecek, Sven Nilsen, 2020
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/adjoint-paths.pdf