# Proving non-existence of monoid symmetric paths

by Sven Nilsen, 2017

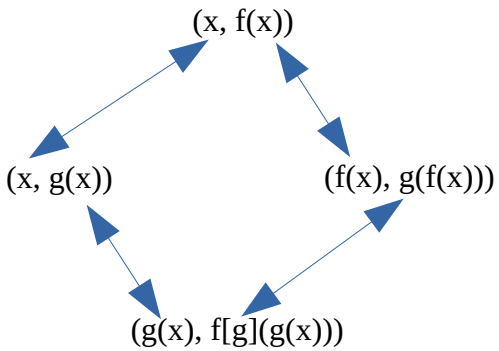
If you have a function `f` and some input `x`, you can construct a pair `(x, f(x))`. Assume you calculate the pair, e.g. `(1, 2)` and forget the original function `f`. Somebody asks you to calculate `f(1)`, and you go looking for `f`, only to find the pair `(1, 2)`. Luckily, this pair works as a partial function, because you know the input is stored in the first slot and the output in the second slot.

A symmetric path has an axiom of path semantics, that there sometimes exists (expressed using the operator "?=") a function "out there", called `f[g]` that can predict some property defined by a function `g`, of another function `f`:

f[g](g(x)) ?= g(f(x))

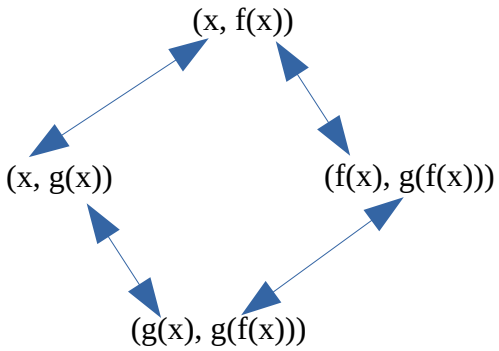
Many fields in mathematics has a symmetric path at its core, e.g. Boolean algebra with DeMorgan's laws. What makes path semantics special is that we can talk about the existence of such functions without knowing what it is, or we can talk about "black boxes" that has properties of known functions. This forms a naturally occurring abstract space, called "path semantical space". The word "path" is used to describe how two functions are connected though other functions like a bridge.

Drawing the axiom of symmetric paths as partial function pairs using double-arrows as equivalences:




The diagram shows there are no inputs or outputs that are not connected by some equation to another pair.

The insight that led to the new proof technique is this: If you are lazy and do not want to go looking for functions all the time, why not just create a partial function for every input value and define `f[g]` in terms of these partial functions? Drawing a new diagram:




A closer look at this pair:

(g(x), g(f(x)))

Every function must compute the same output for the same input, but `g` might be a function that computes the same output for different values of `x`. For example:

g(1) = 2
g(3) = 2

Constructing these pairs:

(2, g(f(1)))
(2, g(f(3)))

Set up this equation:

g(f(1)) = g(f(3))

If this is true, then it is just fine. If it is false, the partial functions can not belong to the same function. Since there can not be more than one function, but either zero or one, it means if any two objects fail to satisfy the equation, the function `f[g]` does not exist.

Formally:

$$\exists\ i, j\ \{\ (\ g(x_i) = g(x_j)\ ) \wedge (\ g(f(x_i)) \neg= g(f(x_j))\ )\ \} \Rightarrow \neg\exists\ f[g]$$

This proof is only valid for functions that take a single argument, not symmetric path that applies `g` to more than one argument. Therefore, the function `f` is in the monoid category.