Modeling Functions

by Sven Nilsen, 2020

In this paper I show how to model functions in Path Semantical Logic, using logical NOT as example.

A function `f : A → B` can be modeled as following in Path Semantical Logic[1]:

f=>(a=>b, a(A)=>b(B))

Here, words starting with small letters are level 1 and words starting with big letters are level 0.
The notation `a(A)` means `a=>A` where `A` is at a lower level.
Comma is the same as `∧` (logical AND).

One can prove the following:

(f, a)=>b

When calling `f` with an argument `a`, it produces a value `b`.

One can also prove the following:

(f, a)=>(A=>B)

However, one can **not** prove the following:

(f, a)=>A	These expressions are not provable, which might be thought of as
(f, a)=>B	`f` taking ownership of `a`, plus `b` not owned after returning.

When only the Constrained Implication Theorem[2] is used, one can think of variables as linear types.
One can use the Normal Implication Theorem[3] to share `a(A)`, e.g. modeling types that can be cloned.

The function `f` is a singleton, because both `A` and `B` contains only one element.
Type with singleton elements are useful in proofs of parametricity[4].
Path Semantical Logic is best at modeling proofs where all types are generic[5].
However, one can push the limits of this logic, to reason a little bit about concrete cases.
For example, to model logical NOT, I extend the definition to include cases for both `tr` and `fa`:

∵ notf=>(fa=>tr, tr=>fa, fa(Bool)=>tr(Bool), tr(Bool)=>fa(Bool)
∴ (notf, tr)=>fa, (notf, fa)=>tr, (notf, fa)=>(Bool=>Bool), (notf, tr)=>(Bool=>Bool)

Here, `Bool=>Bool` is a tautology, but one can swap the output type to `BoolOut`.
Similarly, one can swap the output `tr` with `tr_out` and `fa` with `fa_out`:

∵ notf=>(fa=>tr_out, tr=>fa_out, fa(Bool)=>tr_out(BoolOut), tr(Bool)=>fa_out(BoolOut)
∴ (notf, tr)=>fa_out, (notf, fa)=>tr_out
∴ (notf, fa)=>(Bool=>BoolOut), (notf, tr_out)=>(Bool=>BoolOut)

References:

[1] “Path Semantical Logic”
AdvancedResearch, reading sequence on Path Semantics
https://github.com/advancedresearch/path_semantics/blob/master/sequences.md#path-semantical-logic

[2] “Constrained Implication Theorem”
Sven Nilsen, 2020
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/constrained-implication-theorem.pdf

[3] “Normal Implication Theorem”
Sven Nilsen, 2020
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/implication-theorem.pdf

[4] “Parametricity”
Wikipedia
https://en.wikipedia.org/wiki/Parametricity

[5] “Generic Programming”
Wikipedia
https://en.wikipedia.org/wiki/Generic_programming