Negative Types

by Sven Nilsen, 2020

In this paper I show that there is a way of modeling negative types in Path Semantical Logic.

A negative type is a non-existence of some type membership.

In normal Propositional Logic[1], one can prove the following:

> a, A, B:
> a=>A => (a=>(A¬=B))=(a=>¬B)

Since this holds in Propositional Logic, it also holds Path Semantical Logic[2]:

> (a) (A, B):
> a(A) => a(A¬=B)=a(¬B)

Here, the tuple `(a)` has level 1 and the tuple `(A, B)` has level 0.
The notation `a(A)` means `a=>A` where `A` is at a lower level.

The expression `a(A¬=B)` means that `a` is a proof of the types `A` and `B` being unequal.
In Path Semantical Logic, when `a` is a proof of the types `A` and `B` being unequal, it means that no equality can be expressed between any two members of `A` and `B` without leading to a contradiction.

However, since the `a(A¬=B)=a(¬B)` is true in the context of `a(A)`,
it means one can simply write `a(¬B)` in any context where `a` has some type.

In Type Theory[3], one could write `¬(a : B)` to express a negative type.
In Path Semantical Logic, this is written `a(¬B)`.

References:

[1] “Propositional calculus”
Wikipedia
https://en.wikipedia.org/wiki/Propositional_calculus

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

[3] “Type Theory”
Wikipedia
https://en.wikipedia.org/wiki/Type_theory