

# Witness in Path Semantical Logic

by Sven Nilsen, 2020

*In this paper I explain why a witness is required for one-to-many associated propositions.*

Path Semantical Logical separates propositions<sup>[1]</sup> into levels, such that an equality between two propositions in level  $N+1$ , propagates into equality between uniquely associated propositions in level  $N$ . For example, if  $f$  has level  $N+1$  and  $x$  has level  $N$ , then  $f(x)$  associates  $x$  uniquely with  $f$ .

A proposition in Path Semantical Logic can be thought of as a symbol. One is able to associate symbols with something. The identification of symbols leads to identification of what the symbols mean.

When two symbols  $f$  and  $g$  are equal, their associated meaning is also equal:

$$f(x), g(y), f=g \Rightarrow x=y$$

Since  $f=g$ , one might think that the following is true in Path Semantical Logic:

$$f(x), f(y) \Rightarrow x=y$$

However, this is not the case!

The reason for this is that propositions have their own particular semantics, that can be a bit tricky. In order to explain this, it is simpler to start with another example.

The following is a tautology in propositional logic:

$$(f \Rightarrow x) \wedge (f \Rightarrow y) \Rightarrow (f \Rightarrow x=y)$$

Path Semantical Logic uses implication  $\Rightarrow$  under the hood, so one could write this as:

$$f(x), f(y) \Rightarrow f(x=y)$$

Notice that this would be invalid syntax in typed first-order logic.

In typed first-order logic, if  $x : T$  and  $y : T$ , then  $x=y : T=T$ .

The predicate  $f : T \rightarrow \mathbb{B}$  can not take  $T=T$  as argument.

In Sized Type Theory<sup>[2]</sup>, this type error is exploited, overloading with  $f(x \sim y) = (f(x) \sim f(y))$ . However, in general,  $f(x=y)$  would assume that  $f : \mathbb{B} \rightarrow \mathbb{B}$ .

Now, it happens that since all types in propositional logic are  $\mathbb{B}$ , could one make this work? There is a problem though: For all  $x$  and  $y$ , it is not true that  $(x=y)=x$ .

$$(x=y)=x \quad \text{Is **not** a tautology!}$$

Think about it: If  $f(x)$  and  $f(x=y)$ , then all inputs to  $f$  are the same, then  $(x=y)=x$ .

This means that if the following was true in Path Semantical Logic:

$$f(x), f(x=y) \Rightarrow (x=y)=x$$

Then the following would also be true:

$$f(x), f(y) \Rightarrow x=y$$

However, since the first is not natural, the second is also not natural.

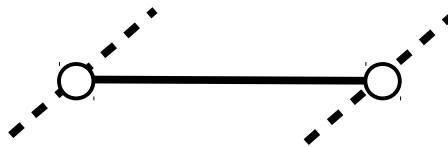
Yet, there is a way out of this problem: Introduce a witness.

To understand witnesses better, one might think of  $f(x)$  and  $f(y)$  as two points in a space.



The space  $f$  is contractible if there a line connecting any two points.

A witness of the line is a surface that intersects with the line:



Instead of allowing a space to be contractible for any line, one can require the existence of a surface intersecting with the line. Any surface will suffice, it just needs to be mentioned explicitly.

However, instead of doing this for every pair of points in space, there is a shortcut: Connect two spaces  $f$  and  $g$  directly, using  $f=g$ .

In Path Semantical Logic, one can prove the following:

$$f(x), f(y), f=g \Rightarrow x=y$$

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

$$f(x), f(x=y), f=g \Rightarrow x=(x=y) \quad \text{Substituting `y` with `x=y`}$$

With other words, substituting a variable  $y$  with  $x=y$  is not sound in Path Semantical Logic.

Yet, one can prove the following:

$$f(x), f(x=y), f=g \Rightarrow x=y$$

As if by magic, Path Semantical Logic seems to know how to destructure equality.

The proofs in this paper were checked by an implementation<sup>[3]</sup> of Path Semantical Logic.

## References:

- [1] “Propositional calculus”  
Wikipedia  
[https://en.wikipedia.org/wiki/Propositional\\_calculus](https://en.wikipedia.org/wiki/Propositional_calculus)
- [2] “Sized Type Theory”  
Sven Nilsen, 2020  
[https://github.com/advancedresearch/path\\_semantics/blob/master/papers-wip/sized-type-theory.pdf](https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/sized-type-theory.pdf)
- [3] “Faster Brute Force Proofs”  
Sven Nilsen, 2020  
[https://github.com/advancedresearch/path\\_semantics/blob/master/papers-wip/faster-brute-force-proofs.pdf](https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/faster-brute-force-proofs.pdf)