Introduction to Path Semantics for Computer Scientists

by Sven Nilsen, 2020-2023

In this paper I give an introduction to Path Semantics written for computer scientists.

Path Semantics[1] is a powerful language for mathematical programming, developed by Sven Nilsen[2].

Here are 3 different ways of thinking about Path Semantics:

- Path Semantics uses a single core axiom[3], which models mathematics in a nutshell
- Path Semantics is a generalized form of dependent types[4]
- Path Semantics is the “inside” of Homotopy Type Theory[5]

There are many ways to learn Path Semantics.
On the website, there are reading sequences[6] organized by subjects.

To explain Path Semantics for computer scientists, one can start with Lambda calculus.

In Lambda calculus[7], there are 3 constructs:

1. Variable `x`
2. Abstraction `\(x) = y` (some authors use `\x.y`)
3. Application `f(x)` (some authors use `f x`)

Together, these 3 constructs form a model of Turing-complete[8] computations.

Path Semantics adds a new construct:

4. Path `[f] y`

These 4 constructs form a model of mathematics (not just computation).
The path construct is the cocategory[9] of the category formed by Application `f(x)`.

By adding paths, one can express mathematical ideas directly with functions that otherwise are only expressible indirectly using equations[10]. Hence, since Path Semantics can express these ideas without variables, it is related to tacit programming[11] and classifying topos[12].

Path Semantics is a result of a general trend in mathematics to move away from set-theoretic foundations[13] of mathematics towards higher-category theory[14]. The inspiration came from:

- A group-theoretic discrete combinatorial generalization of Adinkra diagrams[15]
- Lectures by Vladimir Voevodsky about Univalent foundations[16]

To enforce mathematical consistency, the Path can only be used accordingly to the core axiom[3] of Path Semantics. Currently, it is believed that enumerating all inference rules satisfying to the core axiom is undecidable. For finite models of Boolean algebra, brute force theorem proving confirms the consistency of the theory.

Although paths must satisfy the core axiom, it does not imply the continuum hypothesis being either `true` or `false`[17]. Just a reminder, Path Semantics is not Set Theory, where all mathematical objects needs a model that resembles a set.

The core axiom is purely constructive about the use of symbols, you can define new languages with it, but you can not prove that what you express actually makes sense. It depends, what do you mean by making sense? Hence, Path Semantics supports some ideas of Wittgenstein's view of philosophy of language[18].

The core axiom is not fully formalizable in traditional classical[19] or constructive logic[20] using dependent type prover assistants[21], because:

- Path Semantical Logic[22] is a generalization of classical logic with levels of propositions
- Dependent types is too weak to type check composition in path-space of Path Semantics

Currently, it is believed that type checking in Path Semantics is undecidable, in an even "more undecidable" way than dependent types. Hence, Path Semantics exists in a more expressive universe of mathematical languages.

However, despite extreme expressiveness, by using fixed symbols, one can prove many theorems in Path Semantics safely. This is guaranteed by the core axiom.

Furthermore, by developing a logical foundation of mathematics that extends Propositional Logic, together with an imaginary inverse[23], it is possible to model functional programming that supports dependent types and type checking of composition in path-space. The problems can be overcomed, it just requires research.

References:

[1] "Path Semantics"
AdvancedResearch
https://github.com/advancedresearch/path_semantics

[2] "Sven Nilsen – Github profile"
https://github.com/bvssvni

[3] "Path Semantics"
Sven Nilsen, 2016-2019
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/path-semantics.pdf

[4] "Dependent type"
Wikipedia
https://en.wikipedia.org/wiki/Dependent_type

[5] "Homotopy Type Theory"
https://homotopytypetheory.org/

[6] "Sequences for learning Path Semantics"
AdvancedResearch – Path Semantics
https://github.com/advancedresearch/path_semantics/blob/master/sequences.md

[7] "Lambda calculus"
Wikipedia
https://en.wikipedia.org/wiki/Lambda_calculus

[8] "Turing completeness"
Wikipedia
https://en.wikipedia.org/wiki/Turing_completeness

[9] "cocategory"
nLab
https://ncatlab.org/nlab/show/cocategory

[10] "Normal Paths"
Sven Nilsen, 2019
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/normal-paths.pdf

[11] "Tacit programming"
Wikipedia
https://en.wikipedia.org/wiki/Tacit_programming

[12] "classifying topos"
nLab
https://ncatlab.org/nlab/show/classifying+topos

[13] “Zermelo-Fraenkel set theory”
Wikipedia
https://en.wikipedia.org/wiki/Zermelo%E2%80%93Fraenkel_set_theory

[14] “Higher category theory”
Wikipedia
https://en.wikipedia.org/wiki/Higher_category_theory

[15] “Previous work”
AdvancedResearch – Path Semantics
https://github.com/advancedresearch/path_semantics#previous-work

[16] “Univalent Foundations of Mathematics”
Lectures – Vladimir Voevodsky
https://www.math.ias.edu/vladimir/Univalent_Foundations

[17] “Continuous Lattice Functions”
Sven Nilsen, 2020
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/continuous-lattice-functions.pdf

[18] “Ludwig Wittgenstein”
Stanford Encyclopedia of Philosophy
https://plato.stanford.edu/entries/wittgenstein/

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

[20] “Intuitionistic logic”
Wikipedia
https://en.wikipedia.org/wiki/Intuitionistic_logic

[21] “Calculus of constructions”
Wikipedia
https://en.wikipedia.org/wiki/Calculus_of_constructions

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

[23] “Imaginary Inverse”
Sven Nilsen, 2020
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/imaginary-inverse.pdf