Type theory for programming languages

John D.H. Pritchard [*]

September 13, 2016

Abstract

Need to read Whitehead and Russel and work it back through Hilbert's *Mathematical Logic*. This note reviews a conception of categorical type theory to bring to the literature for review.

1 Objective

Fixing the foundation of a Type system to an abstraction that qualifies and quantifies type design products.

2 Subjective

The application of number theory to type systems was established in Whitehead and Russel. Hilbert's sentential logic puts another stich in that fabric. I haven't found as much as expected that implements type systems relative to these foundations.

We know programming language types as operands and objects, and abstract into the logical members of software systems. We know bit string primitives as the operands of machine instructions. Formulating a continuum from one to the other has been implemented many times for the benefit of practical trade anaylsis. Implementational details aside, what is the foundation of type design analysis and what would a type system notation look like. Why don't we take types seriously. They cost enough.

3 Implementation

The organizational domain maps into the space and time dimensions in functions representing type systems, programming languages, and software systems. That is, the organizational domain is composed of categorical objects of attribute sets representing a type system or a programming language or a software system. An object in the organizational domain has operators for identity and comparison, as well as perhaps composition.

[*]jdp@syntelos.com

4 Conclusion

A type system design (methodology) founded in number theory maximizes language design possibility space extent, and tests or confronts type system design as categorical.

A Notes

The exercise employs the attribute set to encode information into a domain. It describes types as well as spatial information interfaces.