A USABLE FORMAL METHOD FOR IETF SPECIFICATIONS

2024-07-25

Marc Petit-Huguenin marc@petit-huguenin.org

My faith! For more than forty years I have been speaking prose while knowing nothing of it, and I am the most obliged person in the world to you for telling me so.

Le Bourgeois gentilhomme
— Molière

GOAL

The goal of this presentation is to provide the background for the formal-sexpr[1] Internet-Draft.

It has proven difficult to assemble a consistent terminology for this document and this presentation. So the definitions that will be given may differ from the definitions that each person is accustomed with.

---

1. https://datatracker.ietf.org/doc/draft-petithuguenin-ufmrg-formal-sexpr

WHAT IS A PROOF?

A proof is a deductive argument that shows that stated assumptions guarantee a conclusion. It implies the existence of:

- A proposition, which is a declarative statement.
- A process that verifies that the proof is correct for that proposition.

PEN AND PAPER PROOF

- The proposition is in a mix of plain English and mathematical symbols.
- The proof is a sequence of logical steps.
- The verification is done by peer review, i.e., other mathematicians verifying that the sequence of logical steps is correct.

PEN AND PAPER EXAMPLE[1]

Theorem. *Let x be real. If $x^3 - 6x^2 + 11x - 6 = 2x - 2$, then $x = 1$* or $x = 4$.

Proof. By substraction, $x^3 - 6x^2 + 9x - 4 = 0$, which factors as $(x - 1)^2(x - 4) = 0$ ∎

1. Example taken from "99 Variations on a Proof", page 3 by Philip Ording

PROOF ASSISTANTS

The verification step in pen and paper proofs is becoming more and more difficult as the size of the proof grows, and it takes years for errors to be discovered. Proof assistants help with these tasks:

- The proposition is encoded in a logic calculus and the proof is encoded in a formal language.
- The proof is mechanically verified.
- Examples: Isabelle, Tamarin, …

CURRY-HOWARD EQUIVALENCE

Instead of using a proof assistant, we use a strongly typed programming language:

- The proposition is encoded as a dependent type and the proof as code that compiles for that that type.
- type-checking the code is equivalent to verifying the proof for the proposition.
- Examples: Coq, Agda, Lean, F*, Idris2, …

IDRIS2

Idris2[1] is a programming language with a compact syntax inspired by Haskell and a dependent and linear type system.

It uses only ASCII characters, so propositions and proofs can accurately be rendered in the plain-text format of an RFC/I-D, and extracted from the canonical RFCXML format for verification purposes.

1. https://idris2.readthedocs.io/en/latest/

FORMAL-SEXPR

For each section of the original specification the formal-sexpr document provides three subsections:

- An analysis of the text that lists and resolves the ambiguities and contradictions.
- A proposition that follows the text and the analysis and that is encoded as Idris2 types.
- A proof of correctness as Idris2 code for each of the correct examples in the original document.

FORMALIZATION IS HARD

Formalization using a proof assistant or a programming language needs to be very precise because no step is self-evident for a computer, and thus no step can be skipped as is done with pen and paper proofs.

The hard part is that to write a proposition, we need to completely understand what it is about.

FORMALIZATION IS GOOD

This is also what makes that exercise useful for a standard specification, as it is not possible to write the proposition without first resolving all the ambiguities and contradictions.

Even without going through the exercise of proving anything, formalizing a specification has immediate benefits on its quality.

FORMALIZATION IS USEFUL

A formalization can be used beyond ensuring that the text of a specification is correct:

- ABNF Generation
- Diagram Generation
- Specified Test Oracle
- Automatic Testing

QUESTIONS