Computerate Specifying

An introduction

2020-05-15

Marc Petit-Huguenin

History

- Joined IETF in 1998, attending meetings since 2006.
- Worked in RAI and Transport, mostly fixing problems in VoIP related RFCs.
- Interoperability is the reason we are here but incorrect examples in RFCs do not help.

Fixing Stuff

- Provided code to build the examples in RFC 6544.
- Worked on integrating code in AsciiDoc, tooling provided to alpha testers in 2017.
- Realized that code was a partial solution, so started a long quest for a programming language that can also verify code.
- Released draft, tooling, and library using Idris last year.

What is Computerate Specifying

- All formal languages in RFCs are used to be sure that implementations are conform to the RFC.
- Computerate Specifying is about making sure that an RFC is correct in the first place.

How?

- Defining adhoc types for PDUs and State Machines, using the dependent linear type system in Idris.
- Literate programming binds together the code and the document:

```
> trunc : Nat -> String -> String
> trunc l = pack . (take l) . unpack
>
> valid : Int
> valid = current - (rejected + deleted)

But at this point it seems that
{`trunc 5 $ cast $ cast (valid - text) * 100.0 / cast valid`}%
of errata could have been prevented with a more pervasive use of
formal methods.
```

Errata Analysis

- Adding labels to each errata.

example:	Examples could have been correct by construction.
formula:	calculation errors.
language:	Formal languages could have been correct by construction by defining them in a meta-language.

Results

- 25% of errata labeled.

Label	Count	Percentage	
N/A	977	69.09%	
Example	112	7.92%	26%
Formula	118	8.345%	28%
Languages	195	13.7%	46%

Language Results

Label	Count	Percentage
ABNF	71	36.4%
AAD	49	25.1%
ASN.1	40	20.5%
C	13	6.66%

Language Results

Label	Count	Percentage
XML	12	6.15%
Diagram	6	3.07%
TLS	2	1.02%