A quest for an ideal proof language

Dmitry Vlasov

22 июня 2018 г.

QED Manifesto

QED Manifesto - 1994

The goal - to build a computer system/library of formal mathematics with

- rigorous proofs of all theorems
- complete compendium of modern mathematics
- usage as a ligua-franca for mathematicians

QED Manifesto

QED Manifesto - 1994

The goal - to build a computer system/library of formal mathematics with

- rigorous proofs of all theorems
- complete compendium of modern mathematics
- usage as a ligua-franca for mathematicians

Practical applications

- formal verification of programs
- mathematical (and other?) knowledge representation
- (automated) reasoning in expert systems

QED Manifesto FAIL

Passed 20+ years....

QED Manifesto FAIL

Passed 20+ years....

As of 2018

QED project considered to be FAILED

QED Manifesto FAIL

Passed 20+ years....

As of 2018

QED project considered to be FAILED

QED revisited (2007)

An overview paper of F.Wiedijk with critics of most popular/powerful formal math systems

QED Manifesto FAIL

Passed 20+ years....

As of 2018

QED project considered to be FAILED

QED revisited (2007)

An overview paper of F.Wiedijk with critics of most popular/powerful formal math systems

20 years of QED

A 2014 workshop dedicated to the reflection on the success/failures of QED project. Collection of papers in 'Journal of Formalized Mathematics'

Problems with QED-like systems

Types of problems:

- language of expressions (Mizar)
- foundations (HOL, Coq, etc.)
- library organization (all)

Problems with QED-like systems

Types of problems:

- language of expressions (Mizar)
- foundations (HOL, Coq, etc.)
- library organization (all)

"Improving on tradition is good, but ignoring tradition is stupid. Thus, focus in formal mathematics should be on *classical and declarative* systems". F.Wiedijk, 2007

Trust questions [1]

Why we should trust formal proofs?

Especially large formalization of famous theorems

- foundations may be not clear (too complex) - inconsistency?
- implementation may not reflect foundations - bugs?
- implementation language may have vulnerabilities - tricking a system?
- why should we assume good intention of humans?...

[1] M.Adams, Proof Auditing Formalized Mathematics, 2016

Trust questions [1]

Why we should trust formal proofs?

Especially large formalization of famous theorems

- foundations may be not clear (too complex) - inconsistency?
- implementation may not reflect foundations - bugs?
- implementation language may have vulnerabilities - tricking a system?
- why should we assume good intention of humans?...

Foundations for ideal proof language should be extremely simple, at least in translation to some other target language, with complete control of axioms.

[1] M.Adams, Proof Auditing Formalized Mathematics, 2016

Understanding a proof

Proof language readability

A human *should* understand proof:

- naturally
- without external tools (i.e. as is)
- potentially to the ultimate depth

Understanding a proof

Proof language readability

A human *should* understand proof:

- naturally
- without external tools (i.e. as is)
- potentially to the ultimate depth

Proof representation in ideal proof language should be declarative, complete and as close to common mathematicians practice as possible.

QED 2.0 [2]

Shift from rigor to communication

- independence from convention
- independence of content
- dissemination of new results
- modularity and reusability
- organization of knowledge

[2] I.Weiss, The QED Manifesto after two decades - version 2.0, 2016

QED 2.0 [2]

Shift from rigor to communication

- independence from convention
- independence of content
- dissemination of new results
- modularity and reusability
- organization of knowledge

Proof verification is considered optional - dangerous.

[2] I.Weiss, The QED Manifesto after two decades - version 2.0, 2016

QED reloaded [3]

Shift from single-foundation to multi-foundation

- pluralistic approach: no single one foundation
- heterogeneous system
- theory morphisms - a way to interchange knowledge accross different foundations

[3] M.Kohlhase, F. Rabe, QED reloaded: towards a pluralistic formal library of mathematical knowledge, 2016

QED reloaded [3]

Shift from single-foundation to multi-foundation

- pluralistic approach: no single one foundation
- heterogeneous system
- theory morphisms - a way to interchange knowledge accross different foundations

Good intention, but what are the foundations in fact?.. And who is controlling a correctness of theory morphisms?..

[3] M.Kohlhase, F. Rabe, QED reloaded: towards a pluralistic formal library of mathematical knowledge, 2016

Hammering towards QED [4]

Make profit out of moderd ATP

- heavy use of advanced ATP methods
- integration of ATP into ITP
- apply machine learning to ATP in large theories

ATP is really **extremely** important for QED.

What about foundations/reliability of combined 'system'?...

[4] J.Blanchette et al, Hammering towards QED

Russell Logical Framework

Russell

is a *pure* LF, which is a high-level language towards Metamath.

- translates to Metamath, though is not less trustworthy
- uses a declarative, simple and human-readable proof language
- has a flexible syntax of expressions
- type system is very simple

Russell Logical Framework

Russell

is a *pure* LF, which is a high-level language towards Metamath.

- translates to Metamath, though is not less trustworthy
- uses a declarative, simple and human-readable proof language
- has a flexible syntax of expressions
- type system is very simple

Bad news

No special support for rewriting / term reduction / computation

Comparing with Metamath

Russell vs. Metamath

in general the difference is low-level vs. high-level

- explicit definitions (proved conservative)
- explicit grammar rules (CF grammar)
- proof in a purely declarative form (intuitive for a human)
- substitutions are computed by matching and don't litter code

Comparing with Metamath

Russell vs. Metamath

in general the difference is low-level vs. high-level

- explicit definitions (proved conservative)
- explicit grammar rules (CF grammar)
- proof in a purely declarative form (intuitive for a human)
- substitutions are computed by matching and don't litter code

Conclusion

Russell is much more human-friendly then Metamath

Problems with ATP

ATP in Russell

Very problematic.

- (Almost) no way to use commonly used methods
- Extreme combinatorial explosion even in simple cases
- Term reduction is painful

Problems with ATP

ATP in Russell

Very problematic.

- (Almost) no way to use commonly used methods
- Extreme combinatorial explosion even in simple cases
- Term reduction is painful

But...

Although is possible.

Linear method

Linear method

- A directed search, which connects premises with a goal by a chain of inferences *at once*.

- a unit of proving is a *proof tree*, not a single proof tree node
- use ML methods to highlight nodes, which are worth expanding
- is generating proofs in a human manner

Linear method

Linear method

- A directed search, which connects premises with a goal by a chain of inferences *at once*.

- a unit of proving is a *proof tree*, not a single proof tree node
- use ML methods to highlight nodes, which are worth expanding
- is generating proofs in a human manner

GOOD side

- Potentially works good with very large math bases,
- Generates human-like proofs

Linear method

Linear method

- A directed search, which connects premises with a goal by a chain of inferences *at once*.

- a unit of proving is a *proof tree*, not a single proof tree node
- use ML methods to highlight nodes, which are worth expanding
- is generating proofs in a human manner

GOOD side

- Potentially works good with very large math bases,
- Generates human-like proofs

BAD side

- Wouldn't work from scratch - needs substantial proof base for learning
- is not complete in principle

System (implementation)

Russell implementation

Written in c++17 (version no. 3)

- **FAST**
- simple
- reliable
- open source - GPLv3
- https://github.com/dmitry-vlasov/russell

Uses original Metamath math library

System (implementation)

Russell implementation

Written in c++17 (version no. 3)

- **FAST**
- simple
- reliable
- open source - GPLv3
- https://github.com/dmitry-vlasov/russell

Uses original Metamath math library

But...

Speed tradeoffs - consumes a lot of memory space

IDE (implementation)

IDE for Russell

Based on Kate editor

- **USER-FRIENDLY** - main goal
- efficient and easy navigation in math code
- multy-project
- advanced refactoring (not yet done)
- combined ITP/ATP facilities (not yet done)
- open source - GPLv3
- https://github.com/dmitry-vlasov/kate-russell

Plans

Directions of Russell development

- powerfull proving automation (linear method + ML)
- refactoring of Metamath base type system
- introduce theory interpretations
- import other theorem bases (i.e. Mizar base)
- use Russell as a verification tool for flow functional language, integrate it with flow IDE

Thank you for your attention.