Breaking the one-mind-barrier

Lessons from the Liquid Tensor Experiment

by Johan Commelin

On 5 Dec 2020, Peter Scholze posted a challenge:

On 5 Dec 2020, Peter Scholze posted a challenge:

Check the main theorem of liquid vector spaces

On 5 Dec 2020, Peter Scholze posted a challenge:

Check the main theorem of liquid vector spaces

…on a computer

Credit: https://spongebob.gavinr.com/

Credit: https://twitter.com/Jcrudess/status/1338922029278441483/photo/1

1998 Liquid Tension Experiment

1998 Liquid Tension Experiment

1999 Liquid Tension Experiment 2

1998 Liquid Tension Experiment

1999 Liquid Tension Experiment 2

2020 Dec 05: “Liquid Tensor Experiment”,
Peter Scholze

1998 Liquid Tension Experiment

1999 Liquid Tension Experiment 2

2020 Dec 05: “Liquid Tensor Experiment”,
Peter Scholze

2020 Dec 14: announcement LTE 3

1998 Liquid Tension Experiment

1999 Liquid Tension Experiment 2

2020 Dec 05: “Liquid Tensor Experiment”,
Peter Scholze

2020 Dec 14: announcement LTE 3

2021 Apr 16: release LTE 3

1998 Liquid Tension Experiment

1999 Liquid Tension Experiment 2

2020 Dec 05: “Liquid Tensor Experiment”,
Peter Scholze

2020 Dec 14: announcement LTE 3

2021 Apr 16: release LTE 3

2022 Jul 14: formally verified proof of
the main theorem of liquid vector spaces

My topics for this talk

Formal mathematics and …

Please ask questions at any time!

My topics for this talk

Formal mathematics and …

- large collaborations

Please ask questions at any time!

My topics for this talk

Formal mathematics and …

- large collaborations
- cognitive load

Please ask questions at any time!

My topics for this talk

Formal mathematics and …

- large collaborations
- cognitive load
- spec-driven development

Please ask questions at any time!

My topics for this talk

Formal mathematics and …

- large collaborations
- cognitive load
- spec-driven development
- confidence, trust, and evidence

Please ask questions at any time!

Large collaborations (1)

- Maths is dominated by small collaborations ($\leqslant 3$ authors per project)

Large collaborations (1)

- Maths is dominated by small collaborations ($\leqslant 3$ authors per project)
- Large collaborations are happening: 10-author papers, polymath projects

Large collaborations (1)

- Maths is dominated by small collaborations ($\leqslant 3$ authors per project)
- Large collaborations are happening: 10-author papers, polymath projects
- Technology is facilitating large collaborations: weblogs, mathoverflow

Large collaborations (2)

Formalization projects easily scale to large collaborations:

Large collaborations (2)

Formalization projects easily scale to large collaborations:

- Feit-Thompson theorem (Coq) with ≈ 15 contributors

Large collaborations (2)

Formalization projects easily scale to large collaborations:

- Feit-Thompson theorem (Coq) with ≈ 15 contributors
- Kepler/Flyspeck (HOL) with ≈ 20 contributors

Large collaborations (2)

Formalization projects easily scale to large collaborations:

- Feit-Thompson theorem (Coq) with ≈ 15 contributors
- Kepler/Flyspeck (HOL) with ≈ 20 contributors
- Liquid Tensor Exp. (Lean) with ≈ 15 contributors

Large collaborations (3)

Technology used in LTE:

Result:

Large collaborations (3)

Technology used in LTE:

- Blueprint software links $\LaTeX$ and Lean

Result:

Large collaborations (3)

Technology used in LTE:

- Blueprint software links LaTeX and Lean
- Dependency graph tracks progress

Result:

Large collaborations (3)

Technology used in LTE:

- Blueprint software links LaTeX and Lean
- Dependency graph tracks progress
- Proof checker ensures correctness/compatibility

Result:

Large collaborations (3)

Technology used in LTE:

- Blueprint software links LaTeX and Lean
- Dependency graph tracks progress
- Proof checker ensures correctness/compatibility

Result:

- Contributors can work on subprojects that suit them

Large collaborations (3)

Technology used in LTE:

- Blueprint software links LaTeX and Lean
- Dependency graph tracks progress
- Proof checker ensures correctness/compatibility

Result:

- Contributors can work on subprojects that suit them
- Contributors don't *need* to understand the big picture

Large collaborations (3)

Technology used in LTE:

- Blueprint software links LaTeX and Lean
- Dependency graph tracks progress
- Proof checker ensures correctness/compatibility

Result:

- Contributors can work on subprojects that suit them
- Contributors don't *need* to understand the big picture
- Yet have insurance: everything fits together

Large collaborations (4)

The Liquid Tensor Experiment is joint work with:

- Peter Scholze
- Adam Topaz
- Riccardo Brasca
- Patrick Massot
- Scott Morrison
- Kevin Buzzard
- Bhavik Mehta
- Filippo A.E. Nuccio
- Andrew Yang
- Joël Riou
- Damiano Testa
- Heather Macbeth
- Mario Carneiro
- many others

Cognitive load (1)

There are differences between

- finding/creating a proof
- checking/understanding a proof

Cognitive load (1)

There are differences between

- finding/creating a proof
- checking/understanding a proof

I will focus on “checking/understanding”,

although parts also apply to “finding/creating”.

Cognitive load (2)

Cognitive load arises from raw proof complexity.

But also from keeping track of:

Cognitive load (2)

Cognitive load arises from raw proof complexity.

But also from keeping track of:

- side conditions to lemmas

Cognitive load (2)

Cognitive load arises from raw proof complexity.

But also from keeping track of:

- side conditions to lemmas
- discrepancies between statements and proofs

Cognitive load (2)

Cognitive load arises from raw proof complexity.

But also from keeping track of:

- side conditions to lemmas
- discrepancies between statements and proofs
- tweaking definitions/lemmas

Cognitive load (3)

Formal maths *does not* decrease raw proof complexity.

Cognitive load (3)

Formal maths *does not* decrease raw proof complexity.

But it takes care of keeping track of:

- side conditions to lemmas
- discrepancies between statements and proofs
- tweaking definitions/lemmas

Cognitive load (4)

Result:

Cognitive load (4)

Result:

- Most mental RAM can be spent on the big picture

Cognitive load (4)

Result:

- Most mental RAM can be spent on the big picture
- While working on a lemma:
  focus on the proof complexity of that lemma

Cognitive load (4)

Result:

- Most mental RAM can be spent on the big picture
- While working on a lemma:
  focus on the proof complexity of that lemma
- When the proof of the lemma is done:
  seal it off, and focus returns to the big picture

Cognitive load (5)

Experience from LTE:

Cognitive load (5)

Experience from LTE:

- “one key problem I had when I was trying to find this proof was that I was essentially unable to keep all the objects in my ‘RAM’, and I think the same problem occurs when trying to read the proof” — Scholze

Cognitive load (5)

Experience from LTE:

- "one key problem I had when I was trying to find this proof was that I was essentially unable to keep all the objects in my 'RAM', and I think the same problem occurs when trying to read the proof" — Scholze
- My attempts to understand the pen-and-paper proof all failed dramatically

Cognitive load (5)

Experience from LTE:

- “one key problem I had when I was trying to find this proof was that I was essentially unable to keep all the objects in my ‘RAM’, and I think the same problem occurs when trying to read the proof” — Scholze
- My attempts to understand the pen-and-paper proof all failed dramatically
- !! Lean really was a *proof assistant*

Spec-driven development (1)

I claim that

Spec-driven development (1)

I claim that

- large projects benefit from top-down development

Spec-driven development (1)

I claim that

- large projects benefit from top-down development
- write a skeleton first, fill in details later

Spec-driven development (1)

I claim that

- large projects benefit from top-down development
- write a skeleton first, fill in details later
- this outline helps with grasping proof structure

Spec-driven development (1)

I claim that

- large projects benefit from top-down development
- write a skeleton first, fill in details later
- this outline helps with grasping proof structure
- once again, the proof checker gives insurance

Spec-driven development (2)

Abstract ion is one of the most powerful tools in maths

Spec-driven development (2)

Abstraction is one of the most powerful tools in maths

- while creating a new definition,
  write down its main desired properties

Spec-driven development (2)

Abstraction is one of the most powerful tools in maths

- while creating a new definition,
  write down its main desired properties
- these properties form the *spec* (= specification)

Spec-driven development (2)

Abstraction is one of the most powerful tools in maths

- while creating a new definition,
  write down its main desired properties
- these properties form the *spec* (= specification)
- !! refactoring specs is cheap;
  !! refactoring full libraries is not

Spec-driven development (2)

Abstraction is one of the most powerful tools in maths

- while creating a new definition,
  write down its main desired properties
- these properties form the *spec* (= specification)
- !! refactoring specs is cheap;
  !! refactoring full libraries is not
- work out the details later;
  collaborators can easily join here

Spec-driven development (3)

Experience from LTE:

Spec-driven development (3)

Experience from LTE:

1a Wrote down properties of Breen–Deligne resolutions

Spec-driven development (3)

Experience from LTE:

1a Wrote down properties of Breen–Deligne resolutions

1b Discovered easier object with similar behaviour

Spec-driven development (3)

Experience from LTE:

1a Wrote down properties of Breen–Deligne resolutions

1b Discovered easier object with similar behaviour

2a Key statements written down without proofs
after stubbing out definitions

Spec-driven development (3)

Experience from LTE:

1a Wrote down properties of Breen–Deligne resolutions

1b Discovered easier object with similar behaviour

2a Key statements written down without proofs after stubbing out definitions

2b Several definitions and lemmas were tweaked

Spec-driven development (3)

Experience from LTE:

- 1a Wrote down properties of Breen–Deligne resolutions
- 1b Discovered easier object with similar behaviour

- 2a Key statements written down without proofs after stubbing out definitions
- 2b Several definitions and lemmas were tweaked
- 2c After the dust settled, work on the proofs could be distributed

Spec-driven development (3)

Experience from LTE:

1a Wrote down properties of Breen–Deligne resolutions

1b Discovered easier object with similar behaviour

2a Key statements written down without proofs
after stubbing out definitions

2b Several definitions and lemmas were tweaked

2c After the dust settled,
work on the proofs could be distributed

3 Sometimes large proofs or libraries
still had to be refactored (yes, it was painful)

Confidence, trust, and evidence (1)

"How to believe a machine-checked proof" (Pollack, 97)

Confidence, trust, and evidence (1)

"How to believe a machine-checked proof" (Pollack, 97)

- Trusting the hardware

Confidence, trust, and evidence (1)

"How to believe a machine-checked proof" (Pollack, 97)

- Trusting the hardware
- Trusting the software

Confidence, trust, and evidence (1)

"How to believe a machine-checked proof" (Pollack, 97)

- Trusting the hardware
- Trusting the software
- Trusting the formal statement

Confidence, trust, and evidence (1)

"How to believe a machine-checked proof" (Pollack, 97)

- Trusting the hardware
- Trusting the software
- Trusting the formal statement

Let's assume that we trust the hardware and software.

Confidence, trust, and evidence (2)

How can a casual observer gain confidence

that the following align:

Confidence, trust, and evidence (2)

How can a casual observer gain confidence

that the following align:

- mental model of mathematics

Confidence, trust, and evidence (2)

How can a casual observer gain confidence

that the following align:

- mental model of mathematics
- pen-and-paper representation of mathematics

Confidence, trust, and evidence (2)

How can a casual observer gain confidence

that the following align:

- mental model of mathematics
- pen-and-paper representation of mathematics
- formally verified representation of mathematics

Confidence, trust, and evidence (3)

If the statement is easy (e.g., Fermat's Last Theorem)

Confidence, trust, and evidence (3)

If the statement is easy (e.g., Fermat's Last Theorem)

- check all definitions

Confidence, trust, and evidence (3)

If the statement is easy (e.g., Fermat's Last Theorem)

- check all definitions
- check notations (paranoid)

Confidence, trust, and evidence (3)

If the statement is easy (e.g., Fermat's Last Theorem)

- check all definitions
- check notations (paranoid)
- check for zero-width unicode chars (paranoid++)

Confidence, trust, and evidence (4)

The main statement in LTE relies on

Confidence, trust, and evidence (4)

The main statement in LTE relies on

- > 1000 definitions:
  real numbers, profinite sets, p-Banach spaces,
  condensed abelian groups, derived functors, …

Confidence, trust, and evidence (4)

The main statement in LTE relies on

- > 1000 definitions:
  real numbers, profinite sets, p-Banach spaces,
  condensed abelian groups, derived functors, …
- complicated notation:
  to hide functions and parameters that are "irrelevant"

Confidence, trust, and evidence (4)

The main statement in LTE relies on

- > 1000 definitions:
  real numbers, profinite sets, p-Banach spaces,
  condensed abelian groups, derived functors, …
- complicated notation:
  to hide functions and parameters that are "irrelevant"

Some intermediate statements rely on

- zero-width unicode chars:
  to transparently trigger some automation
  while keeping the statement readable

Confidence, trust, and evidence (5)

Abductive reasoning:

If it looks like a duck,
swims like a duck, and
quacks like a duck,
then it probably is a duck.

Confidence, trust, and evidence (6)

Experience from LTE:

Confidence, trust, and evidence (6)

Experience from LTE:

- Create `examples/` folder with 5 files

Confidence, trust, and evidence (6)

Experience from LTE:

- Create `examples/` folder with 5 files
- Each focus on 1 object that occurs in main statement

Confidence, trust, and evidence (6)

Experience from LTE:

- Create `examples/` folder with 5 files
- Each focus on 1 object that occurs in main statement
- !! Short, well-documented, readable

Confidence, trust, and evidence (6)

Experience from LTE:

- Create `examples/` folder with 5 files
- Each focus on 1 object that occurs in main statement
- !! Short, well-documented, readable
- !! Exhibit the “standard behaviour” of the object

Confidence, trust, and evidence (7)

A *profinite* set is a topological space that is

- compact,
- Hausdorff,
- totally disconnected.

A *morphism* of topological spaces is a continuous function.

A *condensed abelian group* is a pro-étale sheaf

$$\mathrm{ProFin}^{\mathrm{op}} \to \mathrm{Ab}.$$

Conclusion

Formal mathematics helps with ...

- large collaborations
- cognitive load
- spec-driven development
- confidence, trust, and evidence