Lean Standard Library

Joe Hendrix (joe@lean-fro.org)

Std - The Lean Standard Library

Std will provide the definitions, lemmas and tactics that all Lean users can rely on.

- Designed for entire Lean user base
- Excellent beginner experience - Good documentation
- Stable
- Fast
- Verification Focused

The Lean core library is designed for Lean and Std developers.

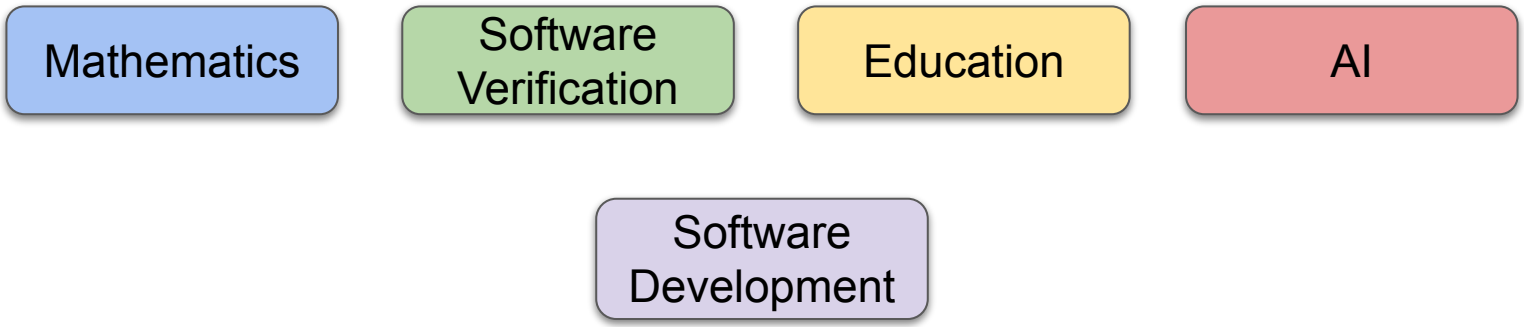
Std is the language for everybody else.

Maintainers and Acknowledgements

I am leading Standard Library development efforts at the Lean Fro.

The community needed a Standard library in 2022, and Mario Carneiro stepped in as maintainer.

Many contributions from Mathlib and other members of the Lean community.



What goes in the Standard Library?

The Standard Library consists of

- Declarations and lemmas on basic types.
- Common data structures with operations and lemmas.
- Type classes and other abstractions
- Operating system operations
- Tactics and commands
  - Manipulation and cleanup tactics
  - Highly automated finishing tactics

Also comprehensive documentation of all user-facing features.

Scope of the Standard Library

For operations and classes, Std will be lean.

- Every operation may need many simplification lemmas.

For lemmas and tactics, Std will be more inclusive:

- Include tactics we think users should know about.
- Include lemmas and search capabilities so users do not have to reprove facts.

Avoid implicitly surprising user:

- `@[simp]` annotations will be restricted to operations that clearly simplify terms.

Key challenge: Verification

- The standard library should provide a consistent experience.
- Contributions should have documentation, testing and verification lemmas:

299 `Std.Data` definitions
1663 `Std.Data` theorems

- Each operation may ultimately require many lemmas.
- Reason to hope automation can help reduce number of user-defined lemmas.

Priorities for 2024

- Standard libraries aligned with overall FRO priorities:

Scalability

Usability

Proof Automation

Documentation

Broad Application in Diverse Fields

- **Tactics** and **contributions** that enable tactics
- Data type and operations used in many projects.
- Broadly useful commands.

Tactics in 2024

- Omega - a linear nat/integer arithmetic over decision procedure
- `exact?` and related tactics.
- Aesop attributes for Std
- A Isabelle Hammer-type tactic that call external solvers.
- Rewriting modulo associativity and commutativity.
- More powerful induction and termination tools.

As tactics evolve, we expect theorems may change or be removed.

Std Documentation

The Lean user manual (under development) will treat Std as part of the language.

- Overview material is just as important as API documentation.
  - Manual will be independent of code, but can pull doc-strings and other content from library.
- Documentation should consider **verifiability**
  - Help people understand normal forms for each type.
  - Are there tactics that could help with library code?
  - Naming and helping users find lemmas will be one of the biggest challenges.
- More details will be forthcoming as documentation tooling is developed.

Stability Goals

Std aims to be a stable library, but 2024 is going to see a lot of changes to make the library more mature.

Possible mitigations:

- Automatic fixes for simple changes when upgrading.
- Selective compilation for libraries to support multiple Lean versions.

Neither is on our immediate roadmap, but may be if they get enough interest from community.

External contributions

External contributions are welcome, but PR reviews may be slow.

- New operations and data types should be well motivated.
  - Priority given to operations need for advanced tactic development.
  - Provide documentation potentially including lemmas.
  - Ensure operation has suitable set of lemmas.
- Tactics are welcome.
  - Recommend tactics first developed in independent repo.
  - Work with Std maintainers to get any dependencies upstream.
  - Evaluate efficacy on larger codebases such as Cedar and Mathlib.