Lax

Édouard Bonnet
CNRS, ENS Lyon, LIP
edouard.bonnet@ens-lyon.fr

Jan Dreier
TU Wien/HPI Postdam
dreier@ac.tuwien.ac.at

Clemens Kuske
TU Wien/HPI Postdam
mail@clemens-kuske.de

Abstract

We introduce lax, an open archive of formalized mathematics. This document outlines what lax is, how you can contribute to it, why you should do so, and what guarantees it provides.

1 What is lax?

Lax is an archive of formalized mathematics, which comprises:

- a command-line interface lax-CLI, which primarily allows one to submit new entries,
- a database, which records the metadata of every submission,
- a website, which enables users to browse these submissions.

The entire project is fully open: the source code of lax-CLI and of the website, the product specification, the database content, the submissions, and the generated artifacts are all public and reusable. Currently, Lean is the only supported language for the submissions, the 'l' of lax stands for Lean, and 'ax' for **arch**ive.

Unlike a library, lax targets contemporary mathematical results, which would take considerable time and effort to be manually formalized. Typically lax proofs are written by AI agents. They are not held to any aesthetic standards,[1] as long as Lean's kernel certifies that they indeed prove what the submission claims. Lax does, however, expect that these claims (the *concepts*, see next section), which constitute the exposed surface on the website, are transparently written and can easily be evaluated and reviewed by humans.

2 What is a lax submission?

A lax submission is a commit to a public Git repository which contains appropriate files and folders. Rather than going through the whole specification, which can be consulted by humans and AI agents, we highlight its salient features.

A lax submission separates its Lean content in two folders, *concepts* and *proofs*. In the concepts folder, each Lean file has a possibly empty list of definitions[2] and a possibly empty list of statements, each introduced as an axiom. The content of the concepts folder is semantically closed: Every declaration in the concepts folder depends only on declarations from that folder and from Mathlib. In the proofs folder, Lean files are unconstrained. In a typical submission, every axiom declared in the concept files is restated in the proofs folder as a theorem and proven (under the basic Lean axioms propext, Quot.sound, Classical.choice).

The content of concept files is exposed on the website. Ideally, the declarations are transparently translated from their natural-language counterparts, and can be understood and reviewed by users

[1] Proofs are where lax intends to be... lax.
[2] introduced with the Lean keywords def, structure, inductive, abbrev, etc.

without a strong Lean or proof-assistant background. In particular, Lean tactics are unwelcome in concept files. Every axiom statement accompanied by a proof (found in the proof package) is marked as proven (green background or THM✓) on the website.

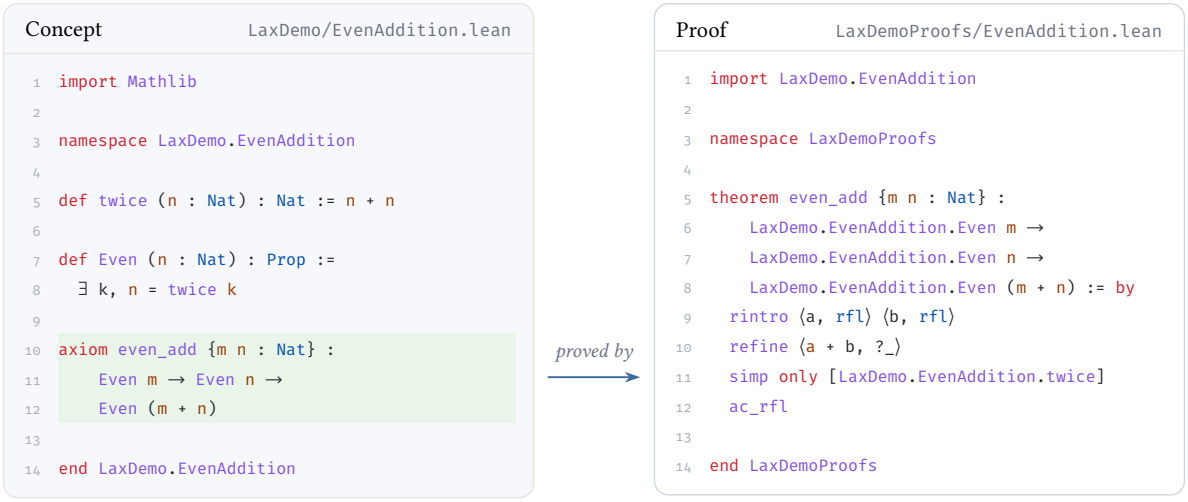

Figure 1. Left: A concept file, as displayed on the website. Right: A matching proof file, as found in the submitter's repository.

By default, a lax submission is initially a *draft*. In this state, it can be modified ad libitum by resubmitting. A submission can be made *permanent*, and then becomes immutable. Lax submissions can reuse and build on past permanent entries.

3 Why use lax?

The starting point is that you are already convinced of the benefits of having results you care about formalized. In 2026, frontier AI agents unlocked the ability of fully autonomously translating natural-language proofs, however long they are, into Lean. Consequently, at a small (and ever smaller) cost, after telling an agent which result to formalize, you may return to the session later and find the requested proof completed.

Once you thus get a Lean proof of, say, your latest result, why make the additional effort of submitting it to lax? Because, this way, your work

- is **more visible**: now a searchable platform (attracting a growing community) points to it.
- is **more legible**: the presentation work is automated away by the submission guidelines, the submitting agent, and the website. The suitable subset of the work that should be exposed is conveniently displayed, with semantic hyperlinks, proof DAGs, concept dependencies, etc.
- can be **more easily built upon**: lax-CLI makes it simple to fetch results from past submissions, for use in completing new Lean projects.
- can be **validated**: users browsing your concept files can vet that those are formalizing the intended mathematical definitions and statements, or otherwise, raise concerns.

Again, AI agents can autonomously and reliably follow lax-CLI's documentation to prepare lax submissions, monitor ongoing submissions, and troubleshoot if need be. The human in the loop mainly checks that the abstract, commented concept files, and optional implementation notes are clean and as intended. This does not take much time. So do not keep your formalized results isolated.

4 Can you trust lax?

If the axiom of a concept is marked as proven on the website, should you trust that the corresponding proof package actually proves it? Assuming you already believe in the soundness of Lean's kernel, you

have to further trust us on basically the entire submission pipeline. This is admittedly too much to ask, even with everything open.

As the Lean code of every submission is publicly available, you can independently verify that

(1) a given project builds and type-checks,

(2) no extra axioms are required by the proof package, and

(3) the type of each axiom in the concept files is definitionally equal to the type of the corresponding theorem in the proof package.

The Lean toolchain can handle (1) and (3), while (2) is done by meta-programming (`#print axioms`). This way you would be reproducing[3] the relevant subset of our submission pipeline, and we would be progressively earning your trust. If you decide to do so, conduct your independent tests in a sandbox. Even if you happen to know the submitter, the Lean code (likely written by an AI agent, and especially so in the proof files) need not be safe.

In most submissions, you should easily be able to tell that the concept files are harmless. Not to complicate the submission process, at this early stage, the pipeline does nothing to guarantee that. Future versions of lax might restrict the concept files to a safe Lean dialect. Users, logged in via ORCID, can vouch that a declaration is formalized correctly, that is, faithful to the mathematical intent, or flag a possible gap (and elaborate in the comment section).

5 How is lax built?

The current lax infrastructure relies on GitHub services:

- the project repositories are hosted on GitHub,
- the submission pipeline uses GitHub Actions,
- the website is built and deployed via GitHub Pages, and
- lax-CLI is available on npm.

Here is a simplified diagram of the lax suite:

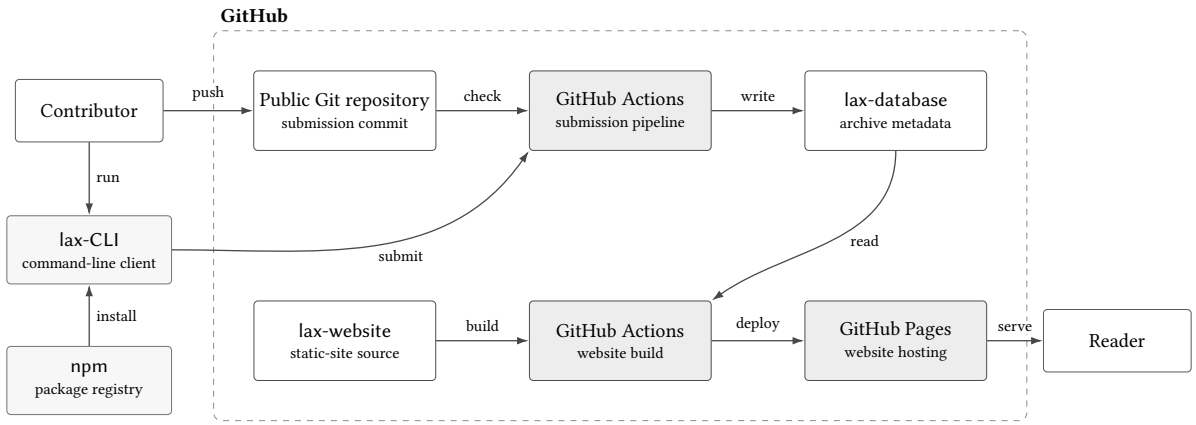

Figure 2. Simplified lax infrastructure. Arrows represent how data flow.

6 How to contribute?

If you are interested in lax, please submit your formalized proofs, review concept files (when that feature is available), talk about the project around you, or reach out to us with comments, inquiries, or suggestions. We are curious to know what you think that we should do next.

[3] Note that the versions of Mathlib, Lean, and Lake used are part of the submission metadata.