An Introduction to Lax

Jan Dreier

September 2026

Abstract

Lax is an archive that links natural-language mathematics with Lean formalizations: an "arXiv for Lean", where submissions build on each other towards a shared body of formalized mathematics. This document is itself a Lax submission, made from the features it describes. A submission consists of *concepts*, *proofs*, and optionally a *paper*, which we introduce in turn, using the prime numbers as a running example.

1 Concepts

Lax represents mathematical definitions and claims as so-called *concepts*. A concept pairs a natural-language statement, as it would appear in a paper, with a faithful encoding of that statement in Lean. This section is annotated with two of them, shown on the right.

Definition 1. A natural number greater than 1 is *prime* if it is divisible only by 1 and by itself.

Theorem 1 (Euclid)**.** *For every natural number n there is a prime $p > n$.*

Even if you are not familiar with Lean, try reading the Lean code to see whether it encodes the intended meaning. One thing stands out: Theorem 1 is stated not as a `theorem` but as an `axiom`, without a proof. That is Lax's way of separating claims from proofs. The correctness of a formal proof is guaranteed by Lean, so with Lax, readers only review the one thing Lean can *not* check: whether the Lean code faithfully represents the intended mathematics. Concepts are therefore written in especially clean and simple Lean code that even non-experts can read. Still, mismatches can be subtle, so it is best to get many pairs of eyes involved. You can click on each concept

to see its full context, including a discussion thread where the faithfulness of the encoding can be debated.

Concepts can be reused across submissions, which lets established academic practices transfer to Lax: submissions build upon each other, and common standards and definitions emerge by an organic selection process. For example, this submission introduces prime numbers as a concept and deliberately leaves one claim about them open (Lemma B below); a follow-up submission can build on it, reuse Definition 1 unchanged, and supply the missing proof. The definition of a prime number is of course already in mathlib [1]. For notions that are not, such as twin-width, nowhere dense graph classes, or the word RAM, Lax can help the community agree on common standards.

2 Proofs

Let us now turn towards proofs, such as this one.

Proof of Theorem 1. Let p be the smallest prime factor of $n! + 1$. If we had $p \leq n$, then p would divide $n!$, and hence also $(n! + 1) - n! = 1$, which is impossible. So $p > n$. □

It is annotated on the right with the corresponding Lax proof. A natural-language proof serves two purposes: it convinces the reader that a claim holds, and it explains why. Lean proofs fully guarantee the former, but are not particularly suited for the latter. They can run to enormous length and are rarely enlightening even when short. Lax therefore treats formal proofs as opaque blobs.

A green badge THM✓ next to a concept indicates that it has a complete formal proof; a yellow one THM× indicates that it does not. However, Lax does not ensure that the formal proof follows the same argument as the prose, or even that the prose is correct. To gain more control over the argument, authors may introduce concepts for the important lemmas along the way, thereby certifying each step of the informal proof rather than only its conclusion.

Lax thereby tracks which lemmas were used to prove which other lemmas. Formally, each proof assumes a set of concepts and derives a target concept. This defines a *proof network* whose nodes are concepts and whose hyperedges are proofs between them. For example, here Lemmas A and B are used to derive Theorem 2.

Lemma A. *Every prime other than* 2 *is odd.*

Proof. Let $p \neq 2$ be prime. If p were even, then 2 would divide p, so by Definition 1 we would have $2 = 1$ or $2 = p$; both are impossible. □

Lemma B (Bertrand's postulate)**.** *For every natural number $n \geq 1$ there is a prime p with $n < p \leq 2n$.*

Theorem 2. *For every natural number $n \geq 2$ there is an odd prime p with $n < p \leq 2n$.*

Proof. By Lemma B there is a prime p with $n < p \leq 2n$. Since $n \geq 2$, we have $p > 2$, so p is odd by Lemma A. □

Note that Lemma B is not proven in this submission. A follow-up submission may provide a proof, thereby completing the proof of Theorem 2, after which both are marked as proven THM ✓ on the website. Clicking on a concept shows its proof network and its open proof obligations. Providing a concept and proving it can thus be separate contributions by separate people.

3 Papers

Besides concepts and proofs, a submission may carry a LaTeX document, like this one. Comments of the form `% lax begin <id>` and `% lax end` annotate the text with Lax objects, where `<id>` names a concept, a proof, or a submission. A normal LaTeX build ignores these comments, so you may upload the same document to arXiv and to Lax. Using Radek Piórkowski's ReflowTeX library [2], the TeX source is rendered for the web as well, reflowed to the reader's screen width beside the as-printed PDF.

4 Submissions

Lean's mathematics traditionally lives in one curated library, mathlib [1]. Lax is a bet on an alternative model: that formal mathematics can grow the way informal mathematics always has, as a literature of independent papers that build on one another without a central authority.

For this to work, submissions must be stable enough to cite. As on arXiv, a registered submission is immutable. This submission, with id `lax-242665`, can be cited via

```
@misc{lax-242665,
  author       = {Dreier, Jan},
```

```
  title        = {An Introduction to Lax},
  year         = {2026},
  howpublished = {Lax, \url{https://laxarchive.org/lax-242665/}}
}
```

This also makes it easy to attach a Lax submission to a conference or journal paper under review; the author list may be left empty if the review process requires anonymity. If changes are needed, you upload a revision under a new id that *supersedes* the old one, and the website points readers from the old version to the new one.

Immutability has a cost on Lax that it does not have on arXiv. A Lean submission can only build on another if both use the same version of mathlib, and mathlib evolves over time. Lax therefore declares a *baseline version* of mathlib (currently `v4.30.0`). Submissions may use any version and are verified and archived either way, but only submissions on the same baseline version can build on each other. The baseline advances rarely. When it does, an older submission is brought forward by a revision, typically by whoever wants to build on it.

5 Writing a submission

The easiest way to get started is the `lax` command-line tool. Install it with

```
npm install -g lax-archive && lax doctor
```

and, inside an empty folder of a public git repository, run `lax init`. This creates the layout the archive expects:

```
manifest.yaml     title, authors, environment, and what it builds on
abstract.md       the summary shown on the submission's page
LICENSE           Apache 2.0, the license of Lean and mathlib
concepts/         a Lean package holding the concepts, one module each
    lakefile.toml
    ...
proofs/           a second Lean package holding the proofs
    lakefile.toml
    ...
```

Next, a prompt like "run `lax print instructions` and formalize the following result" to your favorite coding agent is usually enough to fill these folders with concepts and proofs. This may take several sessions, depending on the scope of the project. Your agent uses `lax build` to make sure

the project is accepted by the archive, and you can run `lax serve` in the project folder to preview a rendering of the work in progress. Once you and your agent are happy, run `lax submit`, which sends the name of the git repository (together with folder and commit hash) to the archive, where it is validated and published. Initially, your submission is in draft state and can be freely overwritten by further submits. When you are ready, run `lax register` to make the submission permanent and citable, so that others can build upon it.

References

[1] The mathlib Community. The Lean Mathematical Library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020)*, pages 367–381, 2020.

[2] Radek Piórkowski. ReflowTeX: reflowable web rendering of LaTeX sources. https://github.com/radek-p/reflowtex, 2026.