Renovating Lean's kernel

Anonymous Author(s)

1 Introduction

Throughout its development for the last decade, Lean's kernel has followed a philosophy of staying small, only adding new features when they were deemed a necessity. Examples of this include the fact that Lean did not have inductive types in Lean 1, no definitional proof-irrelevance until Lean 2, and no structures, nested and mutual inductive types until Lean 4. Despite this trend towards conservativity, Lean has added features which, at the time they were implemented, were uncommon and not well studied. In particular, the presence of an impredicative definitionally proof-irrelevant universe, was found by Coquand [2] to break the strong normalisation property, and thus decidable type-checking of Lean. Since this discovery, much work has been done try and produce a type theory which integrates the feature while still keeping desirable meta-theoretic properties [10–12, 15–17]. Ignoring such type theoretic issues, Lean's kernel also contains implementation bugs. The recent progress in Artifical Intelligence (AI) has lead to the discovery of multiple soundness bugs in the core kernel in the last year [3–9, 13]. In parallel to this, implementation bugs have lead to some incompleteness in the type-checker of Lean, *i.e.*, places where the kernel should accept a term as well-typed but doesn't.

The ingredients provided by recent research are now sufficient for Lean to be able to have decidable type-checking with minimal changes. In light of this, we present existing shortcomings in Lean's kernel, as well as a renovation plan to improve on these issues, and strengten the foundations of the language.

2 Strong normalization

Terms that live in an impredicative proof-irrelevant universe (onwards referred to as `Prop`) may not normalize [2]. However, it is possible to avoid such terms's non-normalization from "leaking" into proof-relevant terms. To do so, it is necessary to ensure proofs never need to be reduced during conversion. Gilbert et al. [12] show that, as long as the elimination criteria for inductive types living in `Prop` is limited enough, proof-relevant terms enjoy strong normalization. However, the elimination principle presented is too limited compared to Lean's, as it does not allow for eliminating the identity type `Eq`, as well as the accessibility predicate `Acc`. Regarding the accessibility predicate, Felicissimo et al. [11] shows that making the type's iota reduction rule be propositional rather than definitional suffices to have decidable type-checking, and that a theory with a definitional such iota rule is conservative over the same theory with its propositional variant. Adding a notion of propositional iota rules to Lean would require non-trivial modifications to its kernel. Instead, we can make the observation that classical axioms such as the global axiom of choice, are sufficient to construct a largely-eliminating recursor for `Acc`, as well as its propositional iota rule, even when the type's base recursor only eliminates to `Prop`. As such, we restrict Lean's kernel to not allow recursive predicates from eliminating to `Type` at all, and recover equivalent recursors with propositional iota using global choice. With these restrictions put in place, the only types that enjoy large elimination out of `Prop` are those that are either structure types whose fields all contain proofs, and "K-like" types, i.e, `Eq`. In both cases, reducing their recursor doesn't require reducing their majors. In the case of structures, eta-expansion suffices to reduce the recursor. For the latter, [10] shows that reducing the recursor for equality can be safely done without the need to reduce its major. One of the two acceptable reduction rule presented in the paper is already implemented in Lean. As such, limiting the elimination of recursive predicates should be enough to ensure proof-relevant terms enjoy strong normalization in Lean.

3 L'état d'η

Conversion algorithms generally come in two flavors: type-directed and untyped. In the former, the type of the two terms getting checked for conversion is known, and used to guide the algorithm. In the latter, that information is not present. Type information is useful when working in type theories containing η rules, as it can be used to ensure terms getting checked are always in η-long form. Untyped conversin can handle most cases of η-conversion, including η for function types, and records that are not "unit types" [1]. Rocq and Idris 2 both rely on an untyped conversion algorithm, avoiding the issue of unit types entirely, while Agda has a typed conversion algorithm which handles all flavors of unit types. Lean sits in the uncomfortable middle ground of having a "relatively" untyped conversion algorithm: it is untyped in the sense that the algorithm doesn't carry the type information, but it may infer types of terms on-the-fly to check for proof-irrelevance and η for unit types. Managing η for unit types in a system with untyped conversions turns out to be non-trivial, and leads to many corner-cases which Lean does not currently handle. Parts of the renovation work includes managing all cases to avoid incompletenesses in the implemented algorithm.

4 Universe polymorphism

Lean's current algorithm for deciding equality between universe levels is incomplete for performance reasons, as this decision problem is coNP-complete [14]. A few (mechanised) complete algorithms now exist, though replacing the current one requires ensuring minimal performance cost for existing codebases.

References

[1] Andreas Abel and Thierry Coquand. 2007. Untyped Algorithmic Equality for Martin-Löf's Logical Framework with Surjective Pairs. *Fundamenta Informaticae* 77, 4 (2007), 345–395. http://fi.mimuw.edu.pl/abs77.html#15 TLCA'05 special issue..

[2] Thierry Coquand. [n. d.]. Canonicity and normalisation for Dependent Type Theory. abs/1810.09367 ([n. d.]). arXiv:1810.09367

[3] Leonardo de Moura. 2026. fix: apply 14807 fix to inductive.h. https://github.com/leanprover/lean4/pull/14843

[4] Leonardo de Moura. 2026. fix: make kernel is_def_eq caching order-independent. https://github.com/leanprover/lean4/pull/14808

[5] Leonardo de Moura. 2026. fix: make the kernel is_prop check require a sort. https://github.com/leanprover/lean4/pull/14807

[6] Leonardo de Moura. 2026. fix: recognize sorts as Prop up to universe normalization. https://github.com/leanprover/lean4/pull/14613

[7] Leonardo de Moura. 2026. fix: reject declarations naming the kernel's nested auxiliary types. https://github.com/leanprover/lean4/pull/14616

[8] Leonardo de Moura. 2026. Postmortem for kernel soundness bug 14576. https://leodemoura.github.io/blog/2026-8-1-postmortem-for-kernel-soundness-bug-14576/

[9] Leonardo de Moura. 2026. Postmortem for the Kernel Soundness Bug Hunt. https://leodemoura.github.io/blog/2026-8-24-postmortem-for-the-kernel-soundness-bug-hunt/

[10] Thiago Felicissimo, Rafael Bocquet, Kenji Maillard, Nicolas Tabareau, Éric Tanter, and Théo Winterhalter. 2026. Consolidating Equality in a Proof Irrelevant Universe. (2026). https://inria.hal.science/hal-05688985 working paper or preprint.

[11] Thiago Felicissimo, Yann Leray, Loïc Pujet, Nicolas Tabareau, Éric Tanter, and Théo Winterhalter. 2026. Definitional Proof Irrelevance Made Accessible. In *41st Annual Symposium on Logic in Computer Science (LICS 2026) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 380)*, Claudia Faggian and Joost-Pieter Katoen (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 41:1–41:26. https://doi.org/10.4230/LIPIcs.LICS.2026.41

[12] Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau. [n. d.]. Definitional Proof-Irrelevance without K. 3, POPL ([n. d.]), 1–28. https://doi.org/10.1145/329031610.1145/3290316

[13] Kiran Gopinathan. 2026. Kernel accepts wrong-structure projections, allowing an axiom-free proof of False. https://github.com/leanprover/lean4/issues/14576

[14] Yoan Géran Mario Carneiro. [n. d.]. *Lean4Lean*. https://github.com/digama0/lean4lean

[15] Loïc Pujet, Yann Leray, and Nicolas Tabareau. [n. d.]. Observational Equality Meets CIC. 47, 2, Article 6 ([n. d.]), 35 pages. https://doi.org/10.1145/3719342

[16] Loïc Pujet and Nicolas Tabareau. [n. d.]. Impredicative Observational Equality. 7, POPL, Article 74 ([n. d.]), 26 pages. https://doi.org/10.1145/3571739

[17] Loïc Pujet and Nicolas Tabareau. 2022. Observational Equality: Now for Good. 6, POPL, Article 32 (2022), 27 pages. https://doi.org/10.1145/3498693