Issue XX: Analytical Type System

М.Е. Сохацький [1]

[1] Національний технічний університет України
ім. Ігоря Сікорського
26 листопада 2017

Анотація

The formalization of mathematical analysis in proof assistants has advanced significantly with systems like Lean and Coq, which have mechanized key results in functional analysis, such as Bochner integration, L^2 spaces, and the theory of distributions. This article introduces Laurent, a novel proof assistant built on MLTT-72, a minimal Martin-Löf Type Theory with Pi and Sigma types, omitting identity types (e.g., `Id`, `J`) in favor of `Prop` predicates and truncated Sigma types. Laurent embeds explicit primitives for calculus, measure theory, and set theory with open sets and topology directly into its core, complemented by a tactics language inspired by Lean, Coq, and recent near tactics. Designed to unify classical and constructive analysis, it targets the mechanization of Laurent Schwartz's Théorie des Distributions and Analyse Mathématique alongside Errett Bishop's Foundations of Constructive Analysis. We present its foundational constructs and demonstrate its application to theorems in sequences, Lebesgue integration, L^2 spaces, and distributions, arguing that its design offers an intuitive yet rigorous approach to analysis, appealing to classical analysts while preserving constructive precision. Laurent emerges as a specialized tool for computational mathematics, advancing the mechanization of functional analysis.

Зміст

1 Introduction to Laurent

The mechanization of mathematical theorems has transformed modern mathematics, enabling rigorous verification of proofs through computational tools known as proof assistants. Systems like Lean and Coq have emerged as leaders in this field, leveraging dependent type theory to formalize a wide range of mathematical domains.

Despite their successes, Lean and Coq often rely on extensive libraries (e.g., Lean's `mathlib` or Coq's Mathematical Components) and general-purpose tactics—such as `ring`, `field`, or `linearith`—that, while effective, can feel detached from the intuitive reasoning of classical analysis. This gap has inspired the development of Laurent, a proof assistant tailored for mathematical analysis, functional analysis, and distribution theory. Laurent integrates explicit primitives for sets, measures, and calculus into its core, paired with a tactics language akin to Lean and Coq, augmented by recent innovations like `near` tactics [1]. This design aims to reflect the spirit of classical mathematics while enabling constructive theorem-proving, offering a specialized tool for researchers in functional analysis.

This article outlines Laurent's architecture and demonstrates its mechanization of classical and constructive theorems, drawing on examples from sequences, Lebesgue integration, and L^2 spaces. We target formal mathematics audience emphasizing computational mathematics and frontier research in functional analysis.

$$
\begin{gathered}
\text{Laurent} := \text{MLTT} \mid \text{CALC} \\
\text{MLTT} := \text{Cosmos} \mid \text{Var} \mid \text{Forall} \mid \text{Exists} \\
\text{CALC} := \text{Base} \mid \text{Set} \mid \text{Q} \mid \text{Mu} \mid \text{Lim} \\
\text{Cosmos} := \mathbf{Prop} \ : \ \mathbf{U_0} \ : \ \mathbf{U_1} \\
\text{Var} := \mathbf{var}\ \text{ident} \mid \mathbf{hole} \\
\text{Forall} := \forall\ \text{ident E E} \mid \lambda\ \text{ident E E} \mid \text{E E} \\
\text{Exists} := \exists\ \text{ident E E} \mid (\text{E}, \text{E}) \mid \text{E}.\mathbf{1} \mid \text{E}.\mathbf{2} \\
\text{Base} := \mathbb{N} \mid \mathbb{Z} \mid \mathbb{Q} \mid \mathbb{R} \mid \mathbb{C} \mid \mathbb{H} \mid \mathbb{O} \mid \mathbb{V}^n \\
\text{Set} := \mathbf{Set} \mid \mathbf{SeqEq} \mid \mathbf{And} \mid \mathbf{Or} \mid \mathbf{Complement} \mid \mathbf{Intersect} \\
\mid \mathbf{Power} \mid \mathbf{Closure} \mid \mathbf{Cardinal} \\
\text{Q} := -/\sim \mid \mathbf{Quot} \mid \mathbf{Lift_Q} \mid \mathbf{Ind_Q} \\
\text{Mu} := \mathbf{mu} \mid \mathbf{Measure} \mid \mathbf{Lebesgue} \mid \mathbf{Bochner} \\
\text{Lim} := \mathbf{Seq} \mid \mathbf{Sup} \mid \mathbf{Inf} \mid \mathbf{Limit} \mid \mathbf{Sum} \mid \mathbf{Union}
\end{gathered}
$$

2 Lean and Coq in Functional Analysis

Lean, developed by Leonardo de Moura, is built on a dependent type theory variant of the Calculus of Inductive Constructions (CIC), with a small inference kernel and strong automation. Its mathematical library, `mathlib`, includes formalizations of Lebesgue measure, Bochner integration, and L^2 spaces, upporting

proofs up to research-level mathematics. Tactics like `norm_num` and `continuity` automate routine steps, though their generality can obscure domain-specific insights.

Both systems, while powerful, prioritize generality over domain-specific efficiency [2]. Laurent addresses this by embedding analysis primitives directly into its core, inspired by recent advancements in near tactics, which enhance proof search with contextual awareness.

3 The Laurent Theorem Prover

Laurent is designed to mechanize theorems in classical and constructive analysis with a focus on functional analysis. Its core is built on dependent types—Pi (functions) and Sigma (pairs)—augmented by explicit primitives for sets, measures, and calculus operations. Unlike Lean and Coq, where such notions are library-defined, Laurent's primitives are native, reducing abstraction overhead and aligning with classical mathematical notation.

3.1 Basic Constructs and Set Theory

Laurent's syntax begins with fundamental types: natural numbers ($\mathbb{N}$), integers ($\mathbb{Z}$), rationals ($\mathbb{Q}$), reals ($\mathbb{R}$), complex numbers ($\mathbb{C}$), quaternions ($\mathbb{H}$), octanions ($\mathbb{O}$) and n-vectors ($\mathbb{V}^n$) all embedded in the core. Sets are first-class objects, defined using lambda abstractions. For example:

```
let set_a : exp =
  Set (Lam ("x", Real,
    RealIneq (Gt, Var "x", Zero)))
```

represents the set $\{x : \mathbb{R} \mid x > 0\}$. Operations like supremum and infimum are built-in:

$$\sup\{x > 0\} = +\infty,$$
$$\inf\{x > 0\} = 0,$$

computed via `Sup set_a` and `Inf set_a`, reflecting the unbounded and bounded-below nature of the positive reals.

3.2 Measure Theory and Integration

Measure theory is central to functional analysis, and Laurent embeds Lebesgue measure as a primitive:

```
let interval_a_b (a : exp) (b : exp) : exp =
  Set (Lam ("x", Real,
   And (RealIneq (Lte, a, Var "x"),
        RealIneq (Lte, Var "x", b))))

let lebesgue_measure (a : exp) (b : exp) : exp =
  Mu (Real, Power (Set Real), Lam ("A", Set Real,
```

```
    If (RealIneq (Lte, a, b),
        RealOps (Minus, b, a),
        Infinity)))
```

This defines $\mu([a,b]) = b - a$ for $a \leq b$, otherwise ∞. The Lebesgue integral is then constructed:

```
let integral_term : exp =
  Lam ("f", Forall ("x", Real, Real), Lam ("a", Real, Lam ("b", Real,
    Lebesgue (Var "f", Mu (Real, Power (Set Real), Lam ("A", Set Real,
      If (And (RealIneq (Lte, Var "a", Var "b"),
               SetEq (Var "A", interval_a_b (Var "a") (Var "b"))),
          RealOps (Minus, Var "b", Var "a"), Zero))),
      interval_a_b (Var "a") (Var "b")))))
```

representing $\int_{[a,b]} f \, d\mu$, with type signature $f, a, b : \mathbb{R} \to \mathbb{R}$.

3.3 L^2 Spaces

The L^2 space, critical in functional analysis, is defined as:

```
let l2_space : exp =
  Lam ("f", Forall ("x", Real, Real),
    RealIneq (Lt,
      Lebesgue (Lam ("x", Real,
        RealOps (Pow, RealOps (Abs, App (Var "f", Var "x"), Zero),
        RealOps (Plus, One, One))),
        lebesgue_measure Zero Infinity, interval_a_b Zero Infinity),
      Infinity))
```

This encodes $\{f : \mathbb{R} \to \mathbb{R} \mid \int_0^\infty |f(x)|^2 \, d\mu < \infty\}$, leveraging Laurent's measure and integration primitives.

3.4 Sequences and Limits

Laurent mechanizes classical convergence proofs efficiently. Consider the sequence $a_n = \frac{1}{n}$:

```
let sequence_a : exp =
  Lam ("n", Nat, RealOps (Div, One, NatToReal (Var "n")))

let limit_a : exp =
  Limit (Seq sequence_a, Infinity, Zero,
    Lam ("ε", Real, Lam ("p", RealIneq (Gt, Var "ε", Zero),
      Pair (RealOps (Div, One, Var "ε"),
        Lam ("n", Nat, Lam ("q", RealIneq (Gt, Var "n", Var "N"),
          RealIneq (Lt, RealOps (Abs,
          RealOps (Minus, App (sequence_a, Var "n"), Zero), Zero),
            Var "ε")))))))
```

This proves $\lim_{n \to \infty} \frac{1}{n} = 0$, with $\forall \varepsilon > 0$, $\exists N = \frac{1}{\varepsilon}$ such that $n > N$ implies $\left|\frac{1}{n}\right| < \varepsilon$.

4 Examples of Theorem Mechanization

Laurent's design excels in mechanizing foundational theorems across differential calculus, integral calculus, and functional analysis. Below, we present a selection of classical results formalized in Laurent, showcasing its explicit primitives and constructive capabilities.

4.1 Taylor's Theorem with Remainder

Taylor's Theorem provides an approximation of a function near a point using its derivatives. If $f : \mathbb{R} \to \mathbb{R}$ is n-times differentiable at a, then:

$$f(x) = \sum_{k=0}^{n-1} \frac{f^{(k)}(a)}{k!}(x-a)^k + R_n(x),$$

where $R_n(x) = o((x-a)^{n-1})$ as $x \to a$.

In Laurent this encodes the theorem's structure, with diff_k representing the k-th derivative and 'remainder' satisfying the little-o condition, verifiable via Laurent's limit primitives.

4.2 Fundamental Theorem of Calculus

The Fundamental Theorem of Calculus links differentiation and integration. If f is continuous on $[a, b]$, then $F(x) = \int_a^x f(t)\,dt$ is differentiable, and $F'(x) = f(x)$:

Laurent's 'Lebesgue' primitive and 'diff' operator directly capture the integral and derivative, aligning with classical intuition.

4.3 Lebesgue Dominated Convergence Theorem

In functional analysis, the Dominated Convergence Theorem ensures integral convergence under domination. If $f_n \to f$ almost everywhere, $|f_n| \leq g$, and $\int g < \infty$, then $\int f_n \to \int f$: This leverages Laurent's sequence and measure primitives, with 'Limit' automating convergence proofs via near tactics.

4.4 Schwartz Kernel Theorem

For distributions, the Schwartz Kernel Theorem states that every continuous bilinear form $B : \mathcal{D}(\mathbb{R}^n) \times \mathcal{D}(\mathbb{R}^m) \to \mathbb{R}$ is represented by a distribution $K \in \mathcal{D}'(\mathbb{R}^n \times \mathbb{R}^m)$ such that $B(\phi, \psi) = \langle K, \phi \otimes \psi \rangle$: This uses Sigma types to pair the kernel K with its defining property, reflecting Laurent's support for advanced functional analysis.

4.5 Banach Space Duality

In Banach spaces, there's a bijection between closed subspaces of X and X^* via annihilators: $A \mapsto A^\perp$, $B \mapsto {}^\perp B$. Laurent formalizes this as:

```
let bijection_theorem = Π (Set Real, ("X",
  If (banach_space (Var "X"),
    And (
      Π (Set (Var "X"), ("A",
        If (closed_subspace (Var "X", Var "A"),
            Id (Set (Var "X"), Var "A", pre_annihilator (Var "X",
                    annihilator (Var "X", Var "A"))), Bool))),
      Π (Set (dual_space (Var "X")), ("B",
        If (closed_subspace (dual_space (Var "X"), Var "B"),
            Id (Set (dual_space (Var "X")), Var "B", annihilator (Var "X",
                    pre_annihilator (Var "X", Var "B"))), Bool))))), Bool)))
```

This showcases Laurent's ability to handle normed spaces and duality, critical in functional analysis.

4.6 Banach-Steinhaus Theorem

The Banach-Steinhaus Theorem ensures uniform boundedness of operators.

If $\sup_{\alpha \in A} \|T_\alpha x\|_Y < \infty$ for all $x \in X$, then there exists M such that $\|T_\alpha\|_{X \to Y} \leq M$:

This uses Laurent's norm and operator primitives, with near tactics simplifying boundedness proofs.

4.7 de Rham Theorem

The de Rham Theorem relates differential forms and integrals over loops. For an open $\Omega \subset \mathbb{R}^n$ and a C^1 1-form ω, if $\int_\gamma \omega = 0$ for all loops γ, there exists f such that $\omega = df$:

```
let de_rham_theorem =
  Π (Nat, ("n",
    Π (Set (Vec (n, Real, RealOps RPlus, RealOps RMult)), ("Omega",
      Π (one_form Omega n, ("omega",
        And (c1_form Omega n (Var "omega"),
          And (Π (loop Omega n, ("gamma",
              Id (Real, integral (Var "omega", Var "gamma"), zero))),
            Σ (zero_form Omega, ("f", And (
                Id (one_form Omega n, Var "omega", differential (Var "f")),
                Π
(Nat, ("m", If (cm_form Omega n (Var "m") (Var "omega"),
                  cm_form Omega n (Var "m") (Var "f"), Bool))))))))))))
```

This demonstrates Laurent's capacity for topology and differential geometry, integrating forms and limits.

These examples highlight Laurent's versatility, from basic calculus to advanced functional analysis, leveraging its native primitives and tactics for intuitive yet rigorous mechanization.

5 Core Tactics of General Proof Assistant

Laurent's proof assistant leverages a rich tactics language to mechanize theorems in functional analysis, blending classical intuition with constructive rigor. Unlike

general-purpose systems like Lean and Coq, Laurent's tactics are tailored to the domain-specific needs of analysis, incorporating explicit primitives for limits, measures, and algebraic structures. This section outlines key tactics used in Laurent, including specialized solvers for rings, fields, and linear arithmetic, and demonstrates their application to functional analysis proofs.

These tactics form the backbone of proof construction, mirroring Coq's logical framework but optimized for Laurent's syntax.

5.1 Intro

Introduces variables from universal quantifiers. For a goal $\forall x : \mathbb{R}, P(x)$, `intro x` yields a new goal $P(x)$ with x in the context.

5.2 Elim

Eliminates existential quantifiers or applies induction (not fully implemented in the current prototype).

5.3 Apply

Applies a lemma or hypothesis to the current goal (pending full implementation).

5.4 Exists

Provides a witness for an existential quantifier. For $\exists x : \mathbb{R}, P(x)$, `exists 0` substitutes $x = 0$ into $P(x)$.

5.5 Assumption

Closes a goal if it matches a hypothesis or simplifies to a trivial truth (e.g., $0 < \varepsilon$ when $\varepsilon > 0$ is in context).

5.6 Auto

Attempts to resolve goals using context hypotheses, ideal for trivial cases.

5.7 Split

Splits conjunctive goals ($P \wedge Q$) into subgoals P and Q.

6 Analysis-Specific Tactics of Laurent

For functional analysis, Laurent introduces tactics that exploit its calculus and measure primitives. These tactics leverage Laurent's `Limit`, `Lebesgue`, and `RealIneq` primitives, reducing manual effort in limit and integration proofs compared to Lean's library-based approach.

6.1 Limit

Expands limit definitions. For a goal $\lim_{n\to\infty} a_n = L$, it generates:

$$\forall \varepsilon > 0, \exists N : \mathbb{N}, \forall n > N, |a_n - L| < \varepsilon,$$

enabling step-by-step convergence proofs. This is crucial for sequences like $\frac{1}{n} \to 0$.

6.2 Continuity

Unfolds continuity definitions at a point. For a goal `continuous_at (f, a)`, it generates:

$$\forall \varepsilon > 0, \exists \delta > 0, \forall x, |x - a| < \delta \implies |f(x) - f(a)| < \varepsilon,$$

transforming the target into an ε-δ formulation using Laurent's `RealIneq` primitives for inequalities and `RealOps` for arithmetic operations (e.g., absolute value, subtraction). This facilitates step-by-step proofs of continuity, such as for the Fundamental Theorem of Calculus, by exposing the logical structure directly in the prover's core, contrasting with Lean's reliance on library theorems.

6.3 Near

Introduces a neighborhood assumption. Given a goal involving a point a, `near x a` adds $x_{\text{near}} : \mathbb{R}$ and $\delta_x > 0$ with $|x_{\text{near}} - a| < \delta_x$, facilitating local analysis as in Taylor's Theorem.

6.4 ApplyLocally

Applies a local property (e.g., from a `near` assumption) to simplify the goal, automating steps in proofs like the Schwartz Kernel Theorem.

To handle the algebraic manipulations ubiquitous in functional analysis (e.g., norms, integrals), Laurent incorporates solvers inspired by Lean and Coq:

7 Algebraic Solvers

To handle the algebraic manipulations ubiquitous in functional analysis (e.g., norms, integrals), Laurent incorporates solvers inspired by Lean and Coq.

Lean's `ring` and `linarith` rely on `mathlib`, while Coq's `field` uses library-defined fields. Laurent embeds these solvers in its core, alongside analysis tactics, reducing dependency on external definitions. This design accelerates proofs in L^2 spaces, Banach duality, and distribution theory, aligning with the needs of a mathematical audience exploring frontier research in computational analysis.

7.1 Ring

Solves equalities in commutative rings. For example, it verifies:

$$(f(x) + g(x))^2 = f(x)^2 + 2f(x)g(x) + g(x)^2,$$

using $\mathbb{R}$'s ring structure. This is implemented via normalization and equality checking in Laurent's core.

7.2 Field

Resolves field equalities and inequalities involving division. For $\int_0^\infty |f(x)|^2 \, d\mu < \infty$, `field` simplifies expressions like:

$$\frac{f(x)^2}{g(x)^2} = \left(\frac{f(x)}{g(x)}\right)^2 \quad (g(x) \neq 0),$$

crucial for quotient manipulations in Banach spaces.

7.3 Big Number Normalization

Automates numerical simplification and equality checking for expressions involving rational numbers and basic functions. For a goal like $2+3=5$ or $|\sin(0)| = 0$, it evaluates:

$$\texttt{norm\_num} : e \mapsto r,$$

where e is an expression (e.g., $2/3+1/2$, $\ln(1)$), and r is either a rational number (via OCaml's `Num` library) or an unevaluated symbolic form. It supports operations including addition, subtraction, multiplication, division, exponentiation, absolute value, logarithms, and trigonometric functions, approximating transcendental results to high precision (e.g., 20 decimal places for sin, cos). This tactic is essential for verifying norm computations, such as $\|f\|_2^2 = \int |f(x)|^2 \, dx$, by reducing concrete numerical subgoals in Banach space proofs.

7.4 Inequality Set Predicates

Handles linear arithmetic inequalities. In the Banach-Steinhaus Theorem, it proves:

$$\|T_\alpha x\|_Y \leq M \|x\|_X,$$

by resolving linear constraints over $\mathbb{R}$, integrating seamlessly with `RealIneq` backed by Z3 SMT solver (morally correct for inequalities).

8 Discussion and Future Directions

Laurent has built-in primitives for streamline proofs in measure theory, integration, and L^2 spaces, while its tactics language ensures flexibility. Compared to Lean's library-heavy approach or Coq's constructive focus, Laurent balances

classical intuition with formal precision, making it accessible to analysts accustomed to paper-based reasoning. Future work includes expanding Laurent's tactics repertoire, formalizing advanced theorems (e.g., dominated convergence, distribution theory).

Hosted at [1], Laurent invites community contributions to refine its role in computational mathematics.

9 Conclusion

Laurent represents a specialized advancement in theorem mechanization, tailored for classical and constructive analysis. By embedding analysis primitives and leveraging topological tactics and algebraic solvers, it offers a unique tool for functional analysts, complementing the broader capabilities of Lean and Coq. This work underscores the potential of domain-specific proof assistants in advancing computational mathematics.

Література

[1] Affeldt R., Cohen C., Mahboubi A., Rouhling D., Strub P-Y. *Classical Analysis with Coq*, Coq Workshop 2018, Oxford, UK doi:

[2] Boldo S., Lelay C., Melquiond G. *Formalization of Real Analysis: A Survey of Proof Assistants and Libraries*, Mathematical Structures in Computer Science, 2016, 26 (7), pp.1196-1233. doi:10.1017/S0960129514000437

[3] Schwartz, L. *Analyse Mathématique*, Hermann, Paris, 1967.

[4] Bishop, E. *Foundations of Constructive Analysis*, McGraw-Hill, New York, 1967.

[5] Bridges, D. *Constructive Mathematics: A Foundation for Computable Analysis*, Theoretical Computer Science, 1999, 219 (1-2), pp.95–109.

[6] Booij, A. *Analysis in Univalent Type Theory*, PhD thesis, University of Birmingham, 2020. Available at: `https://etheses.bham.ac.uk/id/eprint/10411/7/Booij2020PhD.pdf`

[8] Ziemer, W. P., Torres, M. *Modern Real Analysis*, Springer, New York, 2017. Available at: `https://www.math.purdue.edu/~torresm/pubs/Modern-real-analysis.pdf`

[1] `https://github.com/groupoid/laurent`