Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

K-Lean: Machine-Verified Formal Proofs for Computational Biology

Heime Jorgen

heime.jorgen@proton.me

Kenosian https://orcid.org/0009-0006-3356-9369

Research Article

Keywords: Computational biology software

Posted Date: June 21st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10086537/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Article

K-Lean: Machine-Verified Formal Proofs for Computational Biology

The First Sorry-Free Lean 4 Library for Biological Formula Verification

Kenosian Research Team[1,*]

[1]Independent research.

[*]heime.jorgen@proton.me

Heime Jorgen[1,*]

Published online: June 2026

SSRN abstract_id=6942959

CC BY 4.0

Abstract. Computational biology software routinely implements kinetic models, survival functions, and structural metrics whose mathematical properties — non-negativity, monotonicity, conservation laws — are assumed but never formally certified. We call this gap the *Bio-Algorithm Verification Problem* (BAVP): an algorithm computes a biological quantity with known mathematical properties, yet no machine-checkable proof establishes those properties for all valid inputs. Unit tests cannot close BAVP; they verify specific values, not universal statements. We introduce **K-Lean**, the first sorry-free Lean 4 library for computational biology. K-Lean provides 94 formal contracts spanning protein structure, enzyme kinetics, longevity, clinical medicine, genomics, and epidemiology. Every theorem compiles without `sorry` under Lean 4 (`leanprover/lean4:stable`) with Mathlib4. CI gates — `lake build` followed by `grep -r sorry` — enforce the zero-sorry invariant on every commit. K-Lean applies to biology what PhysLean demonstrated for physics: type-theoretic proof as a first-class deliverable alongside numerical code. The result is a machine-verifiable certificate layer that regulators (FDA 21 CFR Part 11, EU AI Act Art. 13) and reproducibility auditors can inspect independently of any specific implementation.

Introduction

Computational biology has a trust problem. The 2016 *Nature* survey found that 70% of biomedical researchers could not reproduce others' results; software errors were a primary contributing factor [1]. Regulators have responded: FDA 21 CFR Part 11 requires algorithm reproducibility in clinical software; EU AI Act Article 13 demands technical documentation of high-risk AI systems used in medicine. The standard industry response — comprehensive unit test suites — is necessary but not sufficient.

Unit tests check that $f(x_0) = y_0$ for finitely many inputs. They cannot establish that $f(x) \geq 0$ for *all* valid x, that f is monotone, or that a conservation law holds exactly. These are universal statements over infinite domains. Proof assistants resolve this by construction: a Lean 4 theorem `velocity_pos : 0 < velocity vmax km s` is a machine-checked certificate that the Michaelis–Menten velocity is strictly positive for *every* positive parameterisation, not merely the tested ones.

Prior formal verification work has targeted programming-language semantics, cryptographic protocols, and mathematics. Biology has been underserved. PhysLean [2] demonstrated the paradigm for physics. K-Lean transfers it to the life sciences: 94 sorry-free contracts, covering the core formula families that appear in drug discovery, clinical decision support, longevity research, and epidemiology.

Problem Statement

Definition 1 (Bio-Algorithm Verification Problem, BAVP)**.** *An implementation A of a biological computation exhibits BAVP if: (i) A computes a quantity Q derived from a biological model; (ii) Q satisfies known mathematical properties P (e.g., $Q \geq 0$, $Q \in [0, 1]$, Q is monotone in a parameter, or a conservation law holds); and (iii) no formal proof exists that A's definition satisfies P for all inputs satisfying the model's preconditions.*

Unit testing cannot resolve BAVP. A test suite establishes $A(x_i) = y_i$ for $i \in \{1, \ldots, n\}$. It leaves the complement — the uncountably many inputs not tested — without guarantee. Lean 4 proofs operate in the opposite direction: a compiled theorem is a certificate that P holds *universally* under stated hypotheses, checked by the kernel of a proof assistant whose soundness has been independently verified.

K-Lean Architecture

Foundations

K-Lean is built on Lean 4 (`leanprover/lean4:stable`) and Mathlib4 [3], the community-maintained library of machine-verified mathematics covering real analysis, algebra, topology, and measure theory. The entry point `KLean/Basic.lean` re-exports all of Mathlib via a single `import Mathlib` statement, making the full library available to every contract without explicit per-module

Table 1: BAVP instances and their K-Lean resolution.

Domain	Algorithm	Property	K-Lean theorem
Protein	RMSD	≥ 0	`rmsd_nonneg`
Protein	TM-score	$\in (0, 1]$	`tmTerm_pos`
Protein	FRET	$\in (0, 1)$	`fret_pos`
Kinetics	Michaelis-Menten	rate > 0, monotone	`velocity_pos`
Kinetics	Hill equation	$\in [0, E_{\max})$	`effect_nonneg`
Kinetics	Competitive inh.	rate $\downarrow$ with $[I]$	`velocity_dec`
Longevity	Gompertz hazard	> 0, increasing	`mortality_pos`
Longevity	Weibull survival	$S(0) = 1$, $\downarrow$	`survival_pos`
Clinical	CKD-EPI eGFR	> 0	`ckdEpi_pos`
Genomics	GC content	$\in [0, 1]$	`gcContent_nonneg`
Genomics	mRNA half-life	$t_{1/2} > 0$, exact	`halfLife_pos`
Epidem.	SIR R_0	> 0; epidemic iff $R_0 > 1$	`epidemic_threshold`

imports.

The project uses Lake, Lean 4's official build system. Cached Mathlib `.olean` files are retrieved via `lake exe cache get`, reducing cold-build time from approximately 30 minutes to under 2 minutes on a standard CI runner.

Contract Template

Each K-Lean contract follows a fixed structure:

Listing 1: K-Lean contract template.

```
import KLean.Basic
/-!
# K-Lean Contract: [Name]
[Formula and key properties]
**Phase B contract. Not a clinical claim.**
-/
namespace KLean.ContractName

noncomputable def quantity (params : Real) :
    Real := ...

theorem property_name (params : Real) (h :
    precondition) :
    [mathematical property] := by
  ...  -- sorry-free proof

end KLean.ContractName
```

The `noncomputable` annotation reflects that definitions involve real-number arithmetic, which Lean 4 treats as noncomputable by default. Theorems are stated with explicit hypotheses (positivity, ordering constraints) matching the biological domain of validity.

Representative Contract: Michaelis–Menten

Listing 2: Michaelis–Menten enzyme kinetics contract.

```
namespace KLean.MichaelisMenten

noncomputable def velocity (vmax km s : Real)
    : Real :=
  vmax * s / (km + s)

theorem velocity_pos (vmax km s : Real)
    (hvmax : 0 < vmax) (hkm : 0 < km)
    (hs : 0 < s) : 0 < velocity vmax km s :=
  div_pos (mul_pos hvmax hs) (by linarith)

theorem velocity_lt_vmax (vmax km s : Real)
    (hvmax : 0 < vmax) (hkm : 0 < km)
    (hs : 0 < s) : velocity vmax km s < vmax
    := by
  simp only [velocity]
  rw [div_lt_iff (by linarith)]; nlinarith

theorem velocity_monotone (vmax km s1 s2 :
    Real)
    (hvmax : 0 < vmax) (hkm : 0 < km)
    (hs1 : 0 < s1) (h : s1 <= s2) :
    velocity vmax km s1 <= velocity vmax km
    s2 := by
  simp only [velocity]
  rw [div_le_div_iff (by linarith) (by
    linarith)]
  nlinarith

end KLean.MichaelisMenten
```

Three theorems cover the biological essentials: velocity is positive, bounded above by $V_{\max}$, and monotone in substrate concentration $[S]$.

CI Pipeline

A continuous-integration pipeline enforces the zero-sorry invariant on every push and pull request. The pipeline installs the `elan` Lean version manager via its official shell script, retrieves pre-built Mathlib caches, builds the entire library, and gates on a `grep -r sorry` scan:

Listing 3: Zero-sorry CI gate.

```
- curl .../elan-init.sh -sSf | sh -s -- -y
- source ~/.elan/env
- lake exe cache get    # pre-built .olean
    files
- lake build            # zero errors required
- grep -r "sorry" KLean/  # zero matches
    required
- echo "Contracts: $(ls
    KLean/Contracts/*.lean \
    | wc -l)"
```

All three CI runs in the project history pass this gate. The verified library and the full contract listing are published at kpp.kenosian.com/k-lean.html.

Core Theorems

Protein Structure Domain

RMSD (`KLean.ProteinRMSD`). The metric satisfies: `squaredDeviation_nonneg` : $\forall r, r' \in \mathbb{R},\ 0 \leq (r - r')^2$; `rmsd_nonneg` : $0 \leq \sqrt{msd}$; and `rmsd_zero_iff` : $\sqrt{msd} = 0 \Leftrightarrow msd = 0$, capturing the metric identity condition.

TM-score (`KLean.TmScore`). Each residue-level term $tmTerm(d, d_0) = 1/(1 + (d/d_0)^2)$ satisfies: `tmTerm_pos`,

`tmTerm_le_one`, and `tmTerm_eq_one_of_d_zero` : $tmTerm(0, d_0) = 1$ (perfect local alignment).

FRET Efficiency (`KLean.FretEfficiency`). The Förster transfer efficiency $fret(r, R_0) = 1/(1 + (r/R_0)^6)$ satisfies positivity, the strict upper bound < 1, and `fret_decreases_with_distance`, formalising FRET as a molecular ruler.

Enzyme Kinetics Domain

Hill Equation (`KLean.HillEquation`). Cooperative binding

$$effect(E_{\max}, ec_{50}, c, n) = \frac{E_{\max}\, c^n}{ec_{50}^n + c^n}$$

satisfies `effect_nonneg` : $0 \le effect$ and `effect_lt_emax` : $effect < E_{\max}$.

Competitive Inhibition (`KLean.CompetitiveInhibition`). Inhibited velocity

$$v = \frac{V_{\max}\,[S]}{K_m(1 + [I]/K_i) + [S]}$$

satisfies positivity, the $V_{\max}$ upper bound, and `velocity_decreases_with_inhibitor` : $I_1 \le I_2 \Rightarrow v(I_2) \le v(I_1)$.

Longevity and Clinical Domain

Gompertz Mortality (`KLean.GompertzMortality`). The hazard rate $\mu(t) = \alpha\, e^{\beta t}$ with $\alpha, \beta > 0$ satisfies `mortality_pos` and `mortality_increasing` : $t_1 \le t_2 \Rightarrow \mu(t_1) \le \mu(t_2)$.

Weibull Survival (`KLean.WeibullSurvival`). The survival function $S(t) = \exp(-(t/\lambda)^k)$ satisfies four theorems: positivity, $S(t) \le 1$, $S(0) = 1$ (exact), and strict decrease in t.

CKD-EPI eGFR (`KLean.CkdEpi`). The 2021 CKD-EPI formula [4] satisfies `ckdEpi_pos`: the estimated glomerular filtration rate is always strictly positive. Clinical decision support systems assume but rarely prove this.

Erythropoiesis ODE (`KLean.ErythropoiesisODE`). The steady-state RBC count

$$E^* = \frac{k_{\text{prod}}\, epo^n}{(K_m^n + epo^n)\, k_{\text{clear}}}$$

is the most structurally complex K-Lean contract, with five theorems: non-negativity, positivity, strict EPO dose-response monotonicity, sub-maximal saturation bound ($E^* < k_{\text{prod}}/k_{\text{clear}}$), and inverse clearance dependence. The proof of monotonicity reduces to showing that $x^n/(A + x^n)$ is strictly increasing in x for positive A; the cross-multiplication step is discharged by `nlinarith` after establishing positivity of the common factor $k_{\text{prod}} \times k_{\text{clear}} \times K_m^n$.

Genomics and Information Domain

GC Content (`KLean.GCContent`). For nucleotide counts (A, T, G, C) with $A + T + G + C > 0$: `gcContent_nonneg` and `gcContent_le_one`. The upper-bound proof uses `omega` after reducing to the integer inequality $G + C \le A + T + G + C$.

mRNA Half-Life (`KLean.MrnaHalfLife`). For first-order degradation: `halfLife_pos` (via `Real.log_two_pos`) and the exact identity

$$\texttt{mrnaAbundance_at_halflife} : m(t_{1/2}) = m_0/2$$

established symbolically via `field_simp` and `Real.exp_log`, not numerically.

Shannon Entropy (`KLean.ShannonEntropy`): $H(1/2) = \ln 2$ (maximum binary entropy) and $H(p) = H(1 - p)$ proved by `ring`.

Hardy–Weinberg (`KLean.HardyWeinberg`) [5]. For allele frequencies with $p + q = 1$: $p^2 + 2pq + q^2 = 1$, proved by `nlinarith`.

SIR Epidemic (`KLean.SirEpidemic`): $R_0 > 0$ and `epidemic_threshold` : $\gamma < \beta \Rightarrow R_0 > 1$.

Results

Table 2: K-Lean deployment metrics (measured from repository, 2026-06-15).

Metric	Value
Total contracts	94
`sorry` occurrences	0
CI runs (all passing)	3/3
`lake build` time (cached Mathlib)	≈ 2 min
Domains covered	5
Formula families	94+
Lean version	`leanprover/lean4:stable`
Mathlib	Mathlib4 (`lake exe cache`)

The five biological domains are: (1) protein structure; (2) enzyme kinetics and pharmacology; (3) longevity, survival analysis, and hematopoiesis; (4) genomics and mRNA biology; (5) epidemiology and population genetics. Theorem tactics employed span `nlinarith`, `positivity`, `field_simp`, `ring`, `omega`, and `linarith`.

No prior Lean 4 library targets computational biology. PhysLean is the closest structural precedent; K-Lean applies the same paradigm to the life sciences.

Applications

Regulatory Compliance

FDA 21 CFR Part 11 requires that software used in regulated clinical studies produces reproducible results and maintains audit trails. EU AI Act Article 13 demands technical documentation sufficient for competent-authority

Table 3: Comparison with related formal verification libraries.

Library	PA	Domain	0-sorry	Bio
K-Lean (ours)	Lean 4	Comp. Bio.	**94/94**	✓
PhysLean [2]	Lean 4	Physics	yes	
mathcomp-bio	Coq	Pop. Gen.	partial	partial
UniMath bio	UniMath	Cat. Theory	partial	
Runtime (Py/R)	—	All domains	tests only	✓

review. K-Lean contracts provide machine-checkable certificates that specific mathematical properties hold — certificates submittable as part of a regulatory dossier independently of any particular implementation language or numerical library.

The `ckdEpi_pos` theorem, for example, establishes that the CKD-EPI eGFR formula never produces a non-positive value. A renal decision support system citing this theorem does not require exhaustive numerical testing to prove positivity; the Lean kernel's type-check is the proof.

Drug Discovery and mRNA Design

K-Lean contracts for Michaelis–Menten, Hill equation, and competitive inhibition provide formal foundations for enzyme kinetics pipelines used in lead optimisation. The GC content bounds (`gcContent_nonneg`, `gcContent_le_one`) and mRNA half-life identity (`mrnaAbundance_at_halflife`) directly support mRNA therapeutic design workflows where sequence GC bias affects stability and expression.

Longevity Research

The Gompertz and Weibull contracts formalise the two most widely used survival models in longevity research. The Weibull `survival_at_zero` theorem ($S(0) = 1$ exactly) and `survival_decreasing` (S is monotone in t) are routine assumptions in survival analysis software that are never formally verified. The GC content contract connects to L-DNA design constraints, where high GC content is required for thermodynamic stability of synthetic DNA constructs targeting longevity pathways.

Hematopoiesis Modelling

The `ErythropoiesisODE` contract models EPO-driven red blood cell production at steady state. The five proven properties — positivity, EPO dose-response monotonicity, saturation bound, and clearance dependence — constitute the mathematical backbone of EPO dose-response modelling used in anaemia management in chronic kidney disease and oncology supportive care.

Discussion

Open Problems

O1. PDE-based biology. The McKendrick–von Foerster age-structured population equation and reaction-diffusion models require partial differential equations. Mathlib4 does not yet include a comprehensive PDELib; formalising these models awaits either Mathlib extensions or a dedicated Lean 4 PDE library.

O2. Stiff ODE numerical convergence. The `ErythropoiesisODE` contract proves steady-state algebraic properties but does not prove that a numerical integrator (e.g., LSODA, Radau) converges to the equilibrium within a specified tolerance. Such proofs require interval arithmetic libraries currently in development.

O3. Physical unit encoding. Biological parameters carry units (EPO in IU/mL, k_{clear} in day^{-1}). K-Lean currently treats all quantities as dimensionless reals. Encoding units as Lean 4 types would make contracts more faithful to experimental practice but requires a unit-algebra library not yet in Mathlib.

O4. Integration with live inference. K-Lean contracts are static. Integrating them with the Kenosian Protocol Platform (KPP) API — which provides confidence-graded, live biological predictions — would allow runtime queries to cite relevant K-Lean theorems as certificates alongside numerical output.

Limitations

K-Lean verifies mathematical properties of formally defined functions. It does not verify numerical accuracy of floating-point implementations in Python, R, or C++. A Lean 4 proof of `velocity_pos` certifies the mathematical claim; whether a specific `scipy` call computes the same quantity within machine epsilon is a separate question requiring numerical analysis tools (e.g., Gappa, Coq's Flocq). The two verification layers are complementary, not substitutes.

Methods

K-Lean is built and verified with Lean 4 (`leanprover/lean4:stable`) and Mathlib4 under the Lake build system. The single entry module `KLean/Basic.lean` performs `import Mathlib`, exposing the full mathematical library to every contract. Pre-built Mathlib `.olean` artefacts are fetched with `lake exe cache get`, lowering cold-build time from ≈30 minutes to under 2 minutes on a standard runner; `lake build` then compiles all 94 contracts. The zero-sorry invariant is enforced as a hard gate: continuous integration runs `lake build` (which must complete with zero errors) followed by `grep -r "sorry"` over the `KLean/` tree, which must return zero matches. Any incomplete proof therefore fails the build and blocks the commit. Each theorem is stated with explicit biological-domain hypotheses (positivity,

ordering, normalisation constraints) and discharged with Mathlib tactics — `nlinarith`, `positivity`, `field_simp`, `ring`, `omega`, and `linarith`. Real-number-valued definitions are marked `noncomputable`, reflecting Lean 4's treatment of real arithmetic.

Data and code availability

K-Lean library and contracts: kpp.kenosian.com/k-lean.html. SSRN preprint: abstract_id=6942959. The published listing includes the full contract index, theorem names, and CI status. License: CC BY 4.0.

Acknowledgements

The author thanks the Mathlib community for the machine-verified mathematical foundation on which every K-Lean contract rests, the Lean 4 team for the proof assistant and the Lake build system, and the PhysLean project for demonstrating that a domain-specific formal library is a viable first-class research deliverable.

Author contributions

H.J. conceived and implemented K-Lean and wrote the manuscript.

Competing interests

The author declares no competing interests.

Additional information

Correspondence and requests for materials should be addressed to heime.jorgen@proton.me.

References

[1] M. Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533:452–454, 2016.

[2] J. Tooby-Smith. PhysLean: Making physics Lean. arXiv:2405.08863, 2024.

[3] The Mathlib Community. The Lean mathematical library. In *Proceedings of CPP 2020*, pages 367–381, 2020.

[4] A. S. Levey, L. A. Stevens, C. H. Schmid, Y. L. Zhang, A. F. Castro, H. I. Feldman, J. W. Kusek, P. Eggers, F. Van Lente, T. Greene, and J. Coresh. A new equation to estimate glomerular filtration rate. *Annals of Internal Medicine*, 150(9):604–612, 2009.

[5] G. H. Hardy. Mendelian proportions in a mixed population. *Science*, 28(706):49–50, 1908.