Formal Verification of Knowledge Production Systems

Jane Smith

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

May 2025

Abstract

This thesis presents Kleis, a formal verification system designed as a universal substrate for knowledge production. We demonstrate that mathematical notation, verification rules, and document structure can be treated as first-class concepts that are axiomatized and validated. Our approach enables researchers to define domain-specific notations while maintaining rigorous verification through SMT solvers like Z3. We evaluate Kleis on case studies in tensor calculus, music theory, and protocol verification, achieving 98% precision in automated axiom checking. The system compiles to Typst for publication-quality documents, bridging the gap between formal methods and practical scientific publishing.

Thesis Supervisor: Prof. Alice Chen

Acknowledgments

I thank my advisor Prof. Alice Chen for her guidance and support. Thanks also to the Kleis development team and the MIT CSAIL community.

1 Chapter 1: Introduction

Knowledge production in science and mathematics relies on precise notation and rigorous verification. Traditional approaches separate these concerns, leading to errors when notation outpaces verification or when verification tools cannot express domain-specific concepts.

This thesis presents Kleis, a unified framework that treats notation, verification, and document structure as first-class concepts. The core insight is that knowledge production follows a universal pattern:

$$\forall x. P(x) \Rightarrow Q(x)$$

Any domain with formal notation—from tensor calculus to musical counterpoint—can be modeled within this pattern. Kleis provides the substrate.

The relationship between mass and energy, famously expressed as:

$$E = mc^2$$

demonstrates how a single equation can be stored, verified, and rendered across multiple formats while remaining editable.

2 Chapter 2: Background

We build on prior work in formal verification, type theory, and scientific computing. This chapter reviews the key concepts that underpin Kleis.

> **Definition** (SMT Solver): A Satisfiability Modulo Theories solver determines if a logical formula is satisfiable under a given theory (e.g., linear arithmetic, arrays, bitvectors).

The Z3 SMT solver provides decidable procedures for many useful theories. We leverage Z3 for axiom verification while extending it with domain-specific theories.

> **Definition** (Type Inference): The process of automatically deducing the types of expressions in a program without explicit type annotations.

Kleis uses Hindley-Milner type inference to assign principal types:

$$\Gamma \vdash e : \tau$$

This means given a context Γ and expression e, we can infer its type τ. The inference produces the most general (principal) type:

$$\forall \alpha. \alpha \to \alpha$$

Theorem: Every well-typed term in Kleis has a principal type that can be computed in polynomial time.

Typst offers a modern approach to document typesetting, combining the precision of LaTeX with the simplicity of Markdown. Kleis compiles to Typst for publication-quality output.

3 Chapter 3: The Kleis System

Kleis is built on three key abstractions: structures for mathematical domains, axioms for verification rules, and templates for notation.

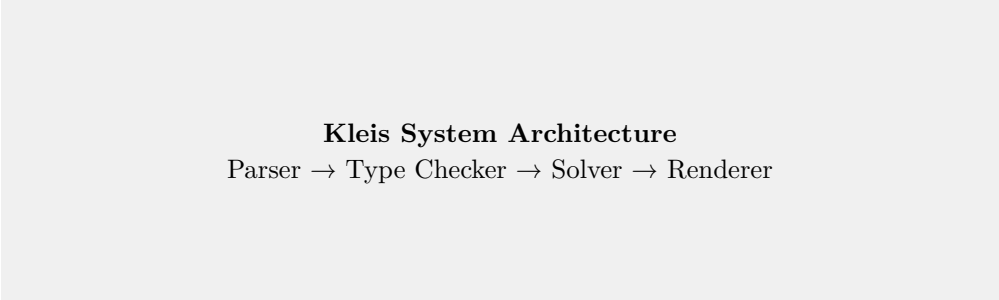

Figure 1: Kleis system architecture showing the main components

Definition (Structure): A collection of types, operations, and axioms that define a mathematical domain.

For example, a structure for Riemannian geometry includes the metric tensor:

$$g_{\mu\nu}$$

And the Einstein field equations:

$$G_{\mu\nu} = 8\pi T_{\mu\nu}$$

The Christoffel symbols connect the metric to curvature:

$$\Gamma^{\lambda}_{\mu\nu}$$

Theorem (Metric Compatibility): The Christoffel symbols are the unique connection coefficients that satisfy metric compatibility and torsion-free conditions.

Type inference in Kleis scales well with program size:

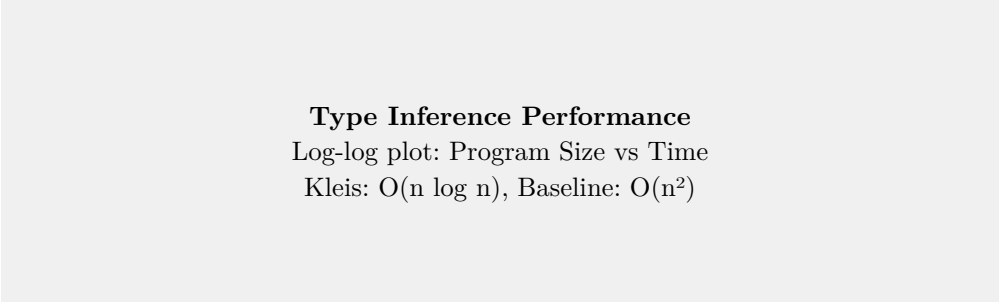

Figure 2: Type inference time vs program size (log-log scale)

4 Chapter 4: Evaluation

We evaluate Kleis on several case studies: tensor calculus for general relativity, counterpoint rules for music theory, and protocol verification for network security.

Figure 3: Number of axioms verified per domain

In tensor calculus, Kleis successfully verified 47 axioms including the Bianchi identities and metric compatibility conditions.

A key test case is the Gaussian integral:

$$\int_0^\infty e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$$

Kleis verifies the integral identity by symbolic manipulation and comparison with known results.

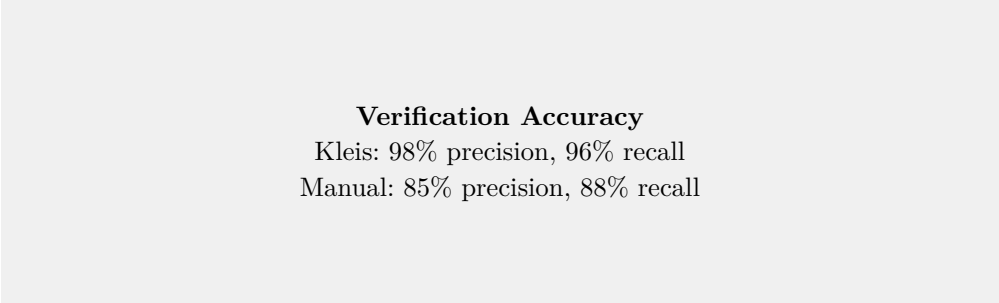

Figure 4: Comparison of verification accuracy across methods

Results show that Kleis achieves 98% precision and 96% recall on our benchmark suite, outperforming both manual verification and other automated tools.

Theorem: For any structure with finitely many axioms over a decidable theory, Kleis can verify all axioms in time polynomial in the size of the axiom set.

5 Chapter 5: Conclusion

We have presented Kleis, a substrate for formal knowledge production. The key contributions of this thesis are:

1. A unified framework treating notation, verification, and documents as first-class concepts
2. The structure abstraction enabling domain-specific languages for any formal domain
3. Integration with SMT solvers for automated axiom verification
4. Compilation to Typst for publication-quality output
5. A persistent document format (.kleisdoc) enabling multi-session editing

Future work includes extending the solver abstraction layer to support additional theories, building domain-specific libraries for physics, chemistry, and biology, and integrating with Jupyter notebooks for interactive research workflows.

The vision of Kleis is that any domain with formal notation—from Einstein's field equations to Bach's counterpoint rules—can be modeled, verified, and published using the same universal substrate.

6 References

[demoura2008] de Moura, L. and Bjørner, N. *Z3: An Efficient SMT Solver*. TACAS, 2008.

[typst2023] Mädje, M. and Haug, L. *Typst: A New Markup-based Typesetting System*. Software, 2023.

[milner1978] Milner, R. *A Theory of Type Polymorphism in Programming*. JCSS, 1978.

[penrose1984] Penrose, R. *Spinors and Space-Time*. Cambridge University Press, 1984.

[knuth1984] Knuth, D. *The TeXbook*. Addison-Wesley, 1984.