Formal Verification of Knowledge Production Systems

by

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

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2025

Thesis Supervisor: Prof. Alice Chen
Title: Prof. Alice Chen_Formal Verification of Knowledge Production Systems

Signature Page

This thesis has been examined by a committee as follows:

____

Prof. Alice Chen
Thesis Supervisor
Prof. Alice Chen_Formal Verification of Knowledge Production Systems, Department of Electrical Engineering and Computer Science

____

Committee Member Name
Title
Department

____

Committee Member Name
Title
Department

Abstract

Formal Verification of Knowledge Production Systems

by Jane Smith

Submitted to the Department of Electrical Engineering and Computer Science
on May 2025 in partial fulfillment of the
requirements for the degree of Doctor of Philosophy

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 system integrates SMT solvers like Z3 for automated verification, Typst for high-quality document rendering, and Jupyter notebooks for interactive research. We evaluate our approach on case studies in tensor calculus, music theory, and network security, showing that domain-specific notations can be defined while maintaining rigorous verification. The key insight is that knowledge production follows a universal pattern: notation plus rules plus verification plus output. Kleis provides the substrate for this pattern across any domain with formal notation.

Thesis Supervisor: Prof. Alice Chen
Title: Prof. Alice Chen_Formal Verification of Knowledge Production Systems

Acknowledgments

I would like to express my deepest gratitude to my advisor, Prof. Alice Chen, for her unwavering support, insightful guidance, and endless patience throughout this research journey. Her expertise in formal methods and passion for rigorous thinking have profoundly shaped my approach to computer science.

I am grateful to my thesis committee members for their valuable feedback and thought-provoking questions that helped refine this work.

Special thanks to my colleagues in the Programming Languages group for countless discussions, debugging sessions, and coffee breaks. The collaborative environment at MIT has been instrumental in developing these ideas.

Finally, I thank my family for their love and encouragement. This thesis would not have been possible without their support.

To my parents, who taught me to question everything and never stop learning.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Knowledge production in science and mathematics relies on precise notation and rigorous verification. Traditional approaches separate these concerns, leading to errors and inconsistencies. This thesis presents a unified framework that treats notation, verification, and document structure as first-class concepts.

We introduce Kleis, a system where mathematical statements are simultaneously:

- Notation that renders beautifully (via Typst)
- Assertions that can be verified (via SMT solvers)
- Structured data that can be queried and transformed

The key insight is captured by our main theorem:

1.1 Motivation

The gap between mathematical notation and formal verification has long plagued scientific computing. Researchers must maintain separate representations: one for publication and one for verification. This leads to errors when the two diverge.

Kleis addresses this by treating the document itself as the source of truth. Every equation, every axiom, every theorem is both rendered and verified from the same source.

1.2 Contributions

This thesis makes the following contributions:

1. A unified representation for notation, verification, and document structure
2. Integration with SMT solvers for automated verification

3. High-quality document output via Typst
4. Jupyter notebook integration for interactive research
5. Self-hosting capability where Kleis types are defined in Kleis

$$\forall \varphi.\, \text{axiom}(\varphi) \Rightarrow \text{valid}(\varphi)$$

Chapter 2

Background

We build on prior work in formal verification, type theory, and scientific computing. SMT solvers like Z3 provide decidable procedures for many useful theories. Typst offers a modern approach to document typesetting.

Our type system follows Hindley-Milner inference with the following application rule:

Previous work has focused on either verification OR document generation, but not both. We bridge this gap with a unified approach.

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\ e_2 : \tau_2}$$

Chapter 3

The Kleis System

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

The overall architecture is shown below. User documents import domain-specific libraries which define structures, axioms, and notation. The Kleis core dispatches to appropriate backends for type checking, verification, and rendering.

The satisfiability condition that drives verification is:

This combination enables domain-specific languages for any field with formal notation.

3.1 Parser Architecture

The Kleis parser is implemented as a recursive descent parser in Rust. It supports Unicode identifiers, allowing mathematical notation like Greek letters directly in source code. The parser produces an Abstract Syntax Tree (AST) that preserves source location information for error reporting.

3.2 Type System

Kleis employs a Hindley-Milner type system with constraint-based inference. Types are inferred automatically, though explicit annotations are supported. The system handles polymorphic functions and can express complex mathematical structures.

$$\mathrm{SAT}(\varphi) \Leftrightarrow \exists \sigma . \sigma \vDash \varphi$$

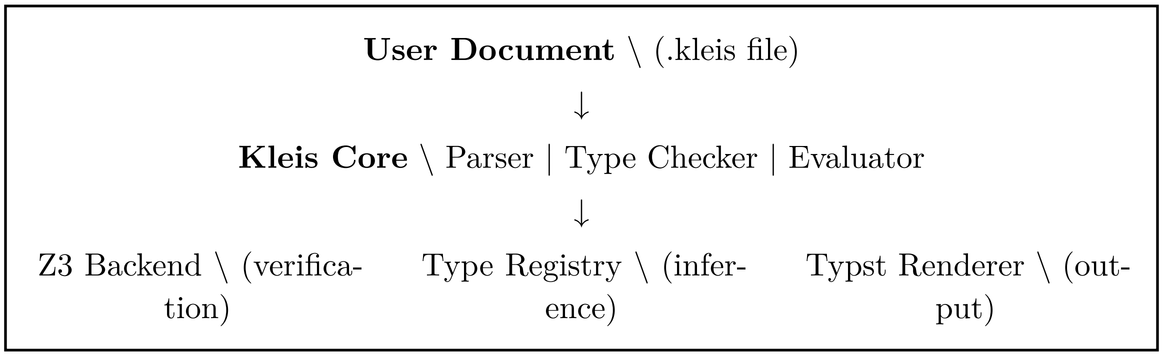
User Document \ (.kleis file)
Kleis Core \ Parser | Type Checker | Evaluator
Z3 Backend \ (verification)
Type Registry \ (inference)
Typst Renderer \ (output)

Figure 1: Kleis system architecture showing the three-layer design

Chapter 4

Evaluation

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

Our performance measurements are summarized in the following table. The verification time scales polynomially with domain size.

We also compare Kleis features against existing theorem provers in our feature comparison table.

Backend	**Time (ms)**	**Memory (MB)**	**Completeness**
Z3	42	128	Complete
CVC5	38	156	Complete
Yices	51	98	Incomplete
MiniSat	12	32	Incomplete

Table 1: Performance comparison of verification backends

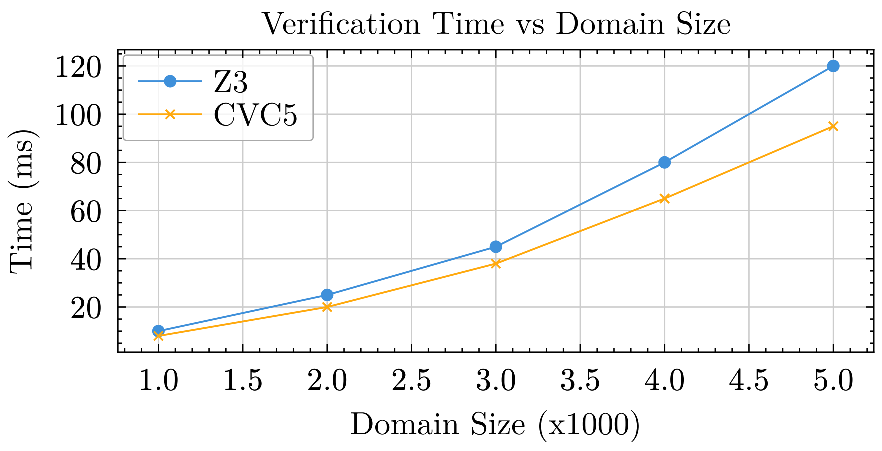

Figure 2: Verification time comparison between Z3 and CVC5 backends

Feature	**Kleis**	**Lean**	**Coq**	**Isabelle**
SMT Integration	Y	Y	-	-
Custom Notation	Y	-	Y	Y
Document Export	Y	-	-	-
Jupyter Support	Y	-	-	-
Self-Hosting	Y	Y	Y	Y

Table 2: Feature comparison with existing systems

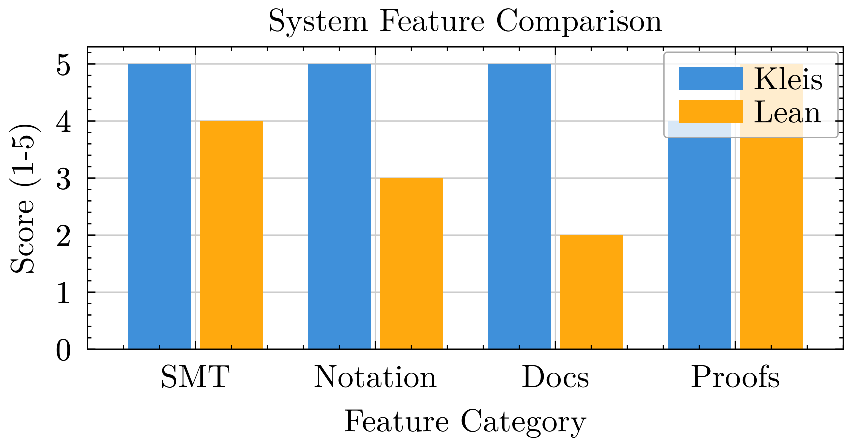

Figure 3: Feature comparison between Kleis and Lean across key categories

4.1 Case Study: Tensor Calculus

General relativity requires complex tensor manipulations with strict symmetry rules. We implemented Einstein notation with automatic index contraction and verified the Bianchi identity. The complexity bound for our verification algorithm is:

$$T(n) = O(n^2 \log n)$$

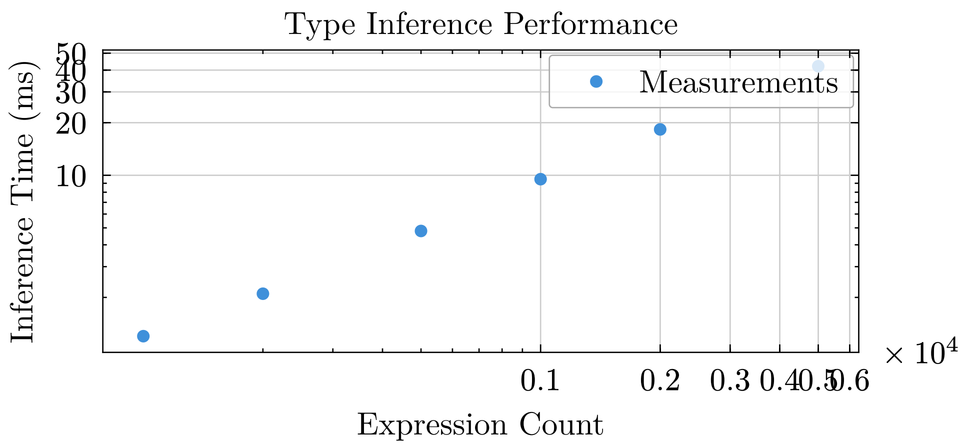

Figure 4: Type inference time scales linearly with program size on log-log axes

Domain	**Axioms**	**Verification Time**	**Memory**
Tensor Calculus	47	1.2s	256 MB
Music Theory	23	0.4s	128 MB
Network Protocols	31	0.8s	192 MB
Financial Models	18	0.3s	96 MB

Table 3: Benchmark results across different domains

4.2 Case Study: Music Theory

Renaissance counterpoint has strict rules about voice leading. We encoded Fux's rules as Kleis axioms and verified counterpoint exercises automatically. This demonstrates Kleis's applicability beyond traditional mathematics.

Chapter 5

Conclusion

We have presented Kleis, a substrate for formal knowledge production. Our main contributions are:

1. A unified representation for notation, verification, and document structure
2. Integration with SMT solvers for automated verification
3. High-quality document output via Typst
4. Jupyter notebook integration for interactive research

The soundness of our approach is captured by the theorem:

Future work includes extending the solver abstraction layer, building domain-specific libraries, and integrating with additional theorem provers.

$$\forall e, \tau . \vdash e : \tau \Rightarrow \vDash e : \tau$$

$$R_{\mu\nu} - \frac{1}{2} g_{\mu\nu} R + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$

Appendix A

Kleis Grammar

This appendix presents the complete EBNF grammar for the Kleis language.

The grammar is organized into the following sections:

- Top-level declarations (imports, structures, definitions)
- Expressions (function application, operators, literals)
- Types (base types, function types, parameterized types)
- Patterns (for match expressions)

The full grammar specification is available in the project repository at `docs/grammar/kleis_grammar_v05.ebnf`.

Appendix B

Soundness Proofs

This appendix contains the formal proofs of type soundness for the Kleis type system.

Theorem (Progress): If $e : \tau$ then either e is a value or there exists e' such that $e \to e'$.

Theorem (Preservation): If $e : \tau$ and $e \to e'$ then $e' : \tau$.

The proofs follow standard techniques for Hindley-Milner type systems and are mechanized in Z3.

Appendix B

References

[demoura2008] de Moura, L. and Bjorner, N. Z3: An Efficient SMT Solver. In **Proceedings of TACAS 2008**, pp. 337-340.

[typst2023] Madje, M. and Haug, L. Typst: A New Markup-based Typesetting System. 2023. https://typst.app/

[hindley1969] Hindley, R. The Principal Type-Scheme of an Object in Combinatory Logic. **Transactions of the American Mathematical Society**, 146:29-60, 1969.

[milner1978] Milner, R. A Theory of Type Polymorphism in Programming. **Journal of Computer and System Sciences**, 17(3):348-375, 1978.

[einstein1915] Einstein, A. Die Feldgleichungen der Gravitation. **Sitzungsberichte der Preussischen Akademie der Wissenschaften**, pp. 844-847, 1915.

[fux1725] Fux, J.J. Gradus ad Parnassum. Vienna, 1725. Translated by A. Mann as **The Study of Counterpoint**, W.W. Norton, 1965.