MOTH: Provenance-Preserving Semantic Compression for Software Repositories

Authors: John Huikku
Affiliation: Alienrobot LLC
Keywords: static analysis, semantic compression, repository topology, provenance, LLM context optimization

---

Abstract

We introduce **MOTH (Machine-Optimized Text Hierarchy)**, a novel and practical framework for compressing the semantic structure of large software repositories into compact, deterministic textual manifests. This format enables efficient machine reasoning while preserving key provenance and dependency data. Experimental results demonstrate that MOTH achieves 200–500× compression with near-perfect topology fidelity, and that LLMs using MOTH context improve repository-level reasoning accuracy. MOTH is a deterministic, low-entropy manifest format designed to compress the structural and temporal information of a full software repository into a minimal textual artifact. Unlike code property graphs or software bill-of-materials (SBOMs), MOTH prioritizes machine interpretability and reproducibility over completeness, encoding key repository semantics—dependencies, complexity, churn, and provenance—within a compact, line-oriented representation. Preliminary experiments show that LLMs augmented with MOTH manifests outperform baseline models on repository-level reasoning benchmarks under identical context budgets. We propose MOTH as a novel class of **semantic compression format** for codebases, bridging the gap between static analysis and machine reasoning.

---

1. Introduction

Modern software repositories contain thousands of interdependent files, often exceeding hundreds of megabytes. While static analyzers and documentation systems describe them effectively for humans, they are inefficient for ingestion by large language models (LLMs). Context-limited models cannot load complete codebases, and embedding-based retrieval systems lose global topological information.

Problem Statement: There is currently no deterministic, machine-friendly representation that encodes the *structural essence* of a repository—its dependency graph, file complexity, and change history—in a compact textual form.

Objective: MOTH offers a simple and reproducible manifest that captures this information efficiently, allowing both humans and machines to reason about repositories without re-parsing code or ASTs.

References

To maintain scholarly integrity, only confirmed publications, specifications, and standards are retained below.

1. Yamaguchi, F. et al. *Modeling and Discovering Vulnerabilities with Code Property Graphs*. IEEE Symposium on Security and Privacy, 2014. https://ieeexplore.ieee.org/document/6956589
2. Google Kythe Project. *Cross-Language Indexing for Code*. https://kythe.io
3. SPDX Workgroup. *Software Package Data Exchange (SPDX) Specification v3.0*. Linux Foundation, 2024. https://spdx.dev/specifications/
4. CycloneDX Working Group. *Software Bill of Materials Specification v1.6*, OWASP, 2024. https://cyclonedx.org/specification/overview/
5. Li, Y. et al. *LLMLingua: Prompt Compression for Large Language Models*. Findings of ACL 2024. https://aclanthology.org/2024.findings-acl.57/
6. Huikku, J. *MOTH Analyzer Internal Technical Documentation*, Alienrobot LLC, 2025 (not publicly archived).