# Systematic Testing for Complex Systems in the Absence of Oracles

Maria Christakis[1]

**Abstract:** Many modern software systems operate in domains where precise specifications are unavailable and reliable test oracles are difficult or impossible to obtain. This is particularly true for systems such as program analyzers, cryptographic proof systems, and machine-learning models, whose correctness depends on complex semantics or learned behavior. This keynote explores how metamorphic testing enables systematic testing in the absence of traditional oracles by checking necessary relations across multiple executions rather than individual outputs. Drawing on experience from testing program analyzers, zero-knowledge proof systems, and machine-learning models, the talk highlights the effectiveness of metamorphic testing across diverse settings. Beyond individual techniques and tools, the keynote distills general lessons on how to reason about correctness when specifications are incomplete, relational, or implicit, and how rigorous testing remains possible even when classical notions of correctness are difficult to apply.

**Keywords:** Metamorphic testing, Test oracles, Program analyzers, Zero-knowledge proof systems, Machine-learning models

## 1 Introduction

Test oracles lie at the heart of software testing: to determine whether a system behaves correctly, one must know the correct behavior. For many of today's software systems, however, such knowledge is incomplete or prohibitively expensive to obtain. This oracle problem arises prominently in domains such as program analysis, cryptographic proof systems, and machine learning, where systems reason about complex semantics, operate over rich mathematical abstractions, or learn behavior from data rather than explicit specifications.

This keynote examines how systematic testing remains possible when traditional test oracles are unavailable. The central thesis is that metamorphic testing provides a unifying foundation for testing complex systems by shifting correctness from individual executions to relations between executions. Instead of predicting exact outputs, metamorphic testing checks whether necessary semantic relationships are preserved across transformations of inputs, programs, or environments. Making this approach effective in practice requires careful design of transformations, relations, and detection mechanisms tailored to each domain.

The talk synthesizes insights from applying metamorphic testing across three classes of systems: program analyzers [Ka24; Ka26], zero-knowledge proof systems [Ho25; HWC26], and machine-learning models [Ch23; En22; EWC24]. Together, these experiences demonstrate the breadth of applicability and challenges of deploying the approach at scale.

[1] TU Wien, Austria, maria.christakis@tuwien.ac.at, https://orcid.org/0000-0002-2649-1958

## 2 Testing Without Oracles in Practice

The oracle problem manifests in different ways across modern software systems. For program analyzers, determining whether an analysis result is correct often requires solving the very problem the analyzer approximates. Our recent work addresses this challenge through interrogation testing, which extends metamorphic testing by exploiting justifications such as counterexamples or invariants to generate related queries that uncover soundness and precision bugs in mature tools [Ka24; Ka26].

In zero-knowledge proof systems, correctness depends on complex multi-stage pipelines, making it difficult to decide whether a proof should be accepted or rejected for a given circuit and input. We applied metamorphic testing by generating semantically equivalent circuit variants and checking for behavioral divergence across the pipeline, revealing soundness and completeness bugs [Ho25; HWC26].

For machine-learning models, the challenge lies in specifying general functional-correctness expectations beyond application-specific metrics. Metamorphic testing captures necessary behavioral constraints by relating multiple executions. Such constraints can be expressed as hyperproperties and checked using automated testing frameworks [Ch23; En22; EWC24].

## 3 Takeaway

Across these domains, many correctness expectations are inherently relational and cannot be captured by single-execution test oracles. Metamorphic testing provides a principled way to express and check such expectations, showing that rigorous testing remains possible when classical notions of correctness are difficult to enforce.

## References

[Ch23] Christakis, M. et al.: Specifying and Testing k-Safety Properties for Machine-Learning Models. In: IJCAI. ijcai.org, pp. 4748–4757, 2023.

[En22] Eniser, H. F. et al.: Metamorphic Relations via Relaxations: An Approach to Obtain Oracles for Action-Policy Testing. In: ISSTA. ACM, pp. 52–63, 2022.

[EWC24] Eniser, H. F.; Wüstholz, V.; Christakis, M.: Automatically Testing Functional Properties of Code Translation Models. In: AAAI. AAAI, pp. 21055–21062, 2024.

[Ho25] Hochrainer, C. et al.: Fuzzing Processing Pipelines for Zero-Knowledge Circuits. In: CCS. ACM, pp. 783–797, 2025.

[HWC26] Hochrainer, C.; Wüstholz, V.; Christakis, M.: Arguzz: Testing zkVMs for Soundness and Completeness Bugs. In: Security. To appear, USENIX, 2026.

[Ka24] Kaindlstorfer, D. et al.: Interrogation Testing of Program Analyzers for Soundness and Precision Issues. In: ASE. ACM, pp. 319–330, 2024.

[Ka26] Kaindlstorfer, D. et al.: Interrogation Testing of CHC Solvers. In: FSE. To appear, ACM, 2026.