A Tale of Two Monads: A Comparative Survey between Freer Monads and MTL

ISAAC H. LOPEZ DIAZ, Department of Computer Science, University of Puerto Rico Rio Piedras, Puerto Rico

Additional Key Words and Phrases: effect systems, monads, algebraic effects, Haskell, interpreters

ACM Reference Format:
Isaac H. Lopez Diaz. 2025. A Tale of Two Monads: A Comparative Survey between Freer Monads and MTL. 1, 1 (December 2025), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction
1.1 Motivation
1.2 Contributions
1.3 Overview
2 Background
2.1 Monads and Computational Effects
2.2 The Expression Problem for Effects
2.3 From Monad Transformers to Algebraic Effects
2.4 The Scheme Subset
3 Effect Systems Under Study
3.1 MTL: Monad Transformer Library
3.1.1 Architecture.
3.1.2 Effect Stack for the Interpreter.
3.1.3 Lifting and Constraints.
3.1.4 Advantages and Limitations.
3.2 freer-simple
3.2.1 Architecture.
3.2.2 Effect Representation.
3.2.3 Handler Composition.
3.2.4 Advantages and Limitations.

Author's Contact Information: Isaac H. Lopez DiazDepartment of Computer Science,, University of Puerto Rico Rio Piedras, Puerto Rico, isaac.lopez@upr.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM XXXX-XXXX/2025/12-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3.3 Polysemy
3.3.1 Architecture.
3.3.2 Effect Representation.
3.3.3 Interpretation Strategies.
3.3.4 GHC Plugin and Optimization.
3.3.5 Advantages and Limitations.

3.4 fused-effects
3.4.1 Architecture.
3.4.2 Effect and Carrier Separation.
3.4.3 Handler Fusion.
3.4.4 Advantages and Limitations.

3.5 effectful
3.5.1 Architecture.
3.5.2 Static vs Dynamic Effects.
3.5.3 Performance Pragmatism.
3.5.4 Advantages and Limitations.

4 Implementation: A Scheme Interpreter
4.1 Abstract Syntax and Shared Infrastructure
4.2 Semantic Domains
4.3 Effects Required
4.3.1 Reader: Lexical Environment.
4.3.2 State: Mutable Bindings and Execution Trace.
4.3.3 Writer: Logging.
4.3.4 Error: Exception Handling.

4.4 MTL Implementation
4.4.1 Type Definitions.
4.4.2 Evaluation Function.
4.4.3 Running the Stack.
4.4.4 Code Observations.

4.5 freer-simple Implementation
4.5.1 Effect Definitions.
4.5.2 Evaluation Function.
4.5.3 Handlers and Interpretation Order.

4.5.4 Code Observations.

4.6 Polysemy Implementation

4.6.1 Effect Definitions.

4.6.2 Evaluation Function.

4.6.3 Interpreters.

4.6.4 Code Observations.

4.7 fused-effects Implementation

4.7.1 Effect and Carrier Definitions.

4.7.2 Evaluation Function.

4.7.3 Handler Composition.

4.7.4 Code Observations.

4.8 effectful Implementation

4.8.1 Effect Definitions.

4.8.2 Evaluation Function.

4.8.3 Dispatching Effects.

4.8.4 Code Observations.

5 Evaluation

5.1 Evaluation Criteria

5.1.1 Performance.

5.1.2 Expressiveness.

5.1.3 Ergonomics.

5.1.4 Ecosystem Integration.

5.2 Benchmark Methodology

5.3 Performance Results

5.3.1 Microbenchmarks.

5.3.2 Interpreter Benchmarks.

5.3.3 Memory Usage.

5.3.4 Compilation Time.

5.4 Expressiveness Comparison

5.4.1 Effect Ordering and Reordering.

5.4.2 Local vs Global Handlers.

5.4.3 Higher-Order Effects.

5.4.4 Effect Polymorphism.

5.5 Ergonomics Comparison

5.5.1 Lines of Code.

5.5.2 Type Signature Complexity.

5.5.3 Error Message Quality.

5.5.4 Refactoring Effort.

6 Discussion

6.1 When to Use Which System

6.2 The Performance–Abstraction Trade-off

6.3 Type System Limitations

6.4 Comparison with Other Languages

6.5 Threats to Validity

7 Related Work

7.1 Effect System Foundations

7.2 Monad Transformers

7.3 Algebraic Effects in Haskell

7.4 Effect System Comparisons

7.5 Interpreters as Effect System Benchmarks

8 Future Work

8.1 Reflective Effect Systems

8.2 Effect Handler Optimization

8.3 Formal Verification of Effect Implementations

9 Conclusion

A Complete Source Code

A.1 Shared Infrastructure

A.2 MTL Implementation

A.3 freer-simple Implementation

A.4 Polysemy Implementation

A.5 fused-effects Implementation

A.6 effectful Implementation

B Benchmark Programs

B.1 Factorial

B.2 Fibonacci

B.3 List Operations

B.4 Environment-Heavy Workloads

B.5 Error-Heavy Workloads

C Raw Benchmark Data