Mnemosyne

A Functional Systems Programming Language

Hawk Weisman

Department of Computer Science
Allegheny College

November 11, 2015

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- But what does that mean?

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- **Functional programming** models computation as the evaluation of functions [4, 7]
  - **It focuses on** immutability, purity, and function composition

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- **Functional programming** models computation as the evaluation of functions [4, 7]
  - **It focuses on** immutability, purity, and function composition
  - **Advantages:** expressiveness [3, 4], modularity [3, 4], safety

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- **Mnemosyne is inspired by:**
  - **Lisp**'s syntax and homoiconicity
  - **Haskell and ML**'s typeclasses, pattern matching and monads
  - **Rust**'s memory management

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- **Systems programming** is the implementation of software that provide services to other software [5, 6].

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- **Systems programming** is the implementation of software that provide services to other software [5, 6].

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- **Systems programming** is the implementation of software that provide services to other software [5, 6].
- High quality systems are necessary for high quality applications.

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- **Systems programming** is the implementation of software that provide services to other software [5, 6].
- High quality systems are necessary for high quality applications.
- But there are some significant challenges in this field [2, 6]

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- Almost all systems programming today is done in C and C++

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- Almost all systems programming today is done in C and C++
- **Why?** C manages memory at compile-time

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- Almost all systems programming today is done in C and C++
- **Why?** C manages memory at compile-time
  - Most languages manage memory through garbage collection (GC) [1]
  - GC is unsuitable for most low-level systems
  - C programmers manage memory manually (`malloc()` and `free()`)

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- **Manual memory management leads to errors** such as buffer overflows, memory leaks, and null pointer dereferences
- **What if there was another way?**

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- Mnemosyne manages memory automatically at compile time
- **How?**

What is Mnemosyne?

A functional systems programming language with compile-time automatic memory management.

- Mnemosyne manages memory automatically at compile time
- **How?**
  - Stack allocation
  - Ownership analysis
  - Controlled mutability

References

David H. Bartley. “Garbage Collection”. In: *Encyclopedia of Computer Science*. Chichester, UK: John Wiley and Sons Ltd., pp. 743–744. ISBN: 0-470-86412-5.

Jim Blandy. *Why Rust?* 1st ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O’Reilly Media, Inc, Sept. 2015. ISBN: 978-1-491-92730-4.

Paul Hudak and Mark P. Jones. *Haskell vs. Ada vs. C++ vs. Awk vs. ... An Experiment in Software Prototyping Productivity*. Research Report YALEU/DCS/RR-1049. New Haven, CT: Department of Computer Science, Yale University, 1994.

John Hughes. “Why functional programming matters”. In: *The Computer Journal* 32.2 (1989), pp. 98–107.

Thomas Narten. “Systems Programming”. In: *Encyclopedia of Computer Science*. Chichester, UK: John Wiley and Sons Ltd., pp. 1739–1741. ISBN: 0-470-86412-5.

Jonathan Shapiro. “Programming Language Challenges in Systems Codes: Why Systems Programmers Still Use C, and What to Do About It”. In: *Proceedings of the 3rd Workshop on Programming Languages and Operating Systems: Linguistic Support for Modern Operating Systems*. PLOS ’06. San Jose, California: ACM, 2006. ISBN: 1-59593-577-0. DOI: 10.1145/1215995.1216004.

David S. Wise. “Functional Programming”. In: *Encyclopedia of Computer Science*. Chichester, UK: John Wiley and Sons Ltd., pp. 736–739. ISBN: 0-470-86412-5.