Type-Safe Modular Hash-Consing Library in Rust and Haskell

BY KARAN AHLUWALIA
ADVISED BY PROF. MATTHEW FLUET

Hash-consing

Strategy for efficiently sharing and storing data that is structurally equal in a purely functional context.

STRUCTURALLY EQUAL IN A PURELY FUNCTIONAL CONTEXT

- "Structurally equal" means that two pieces of data have the same structure or shape, regardless of their specific values.(Binary Trees)

- Functional means that the data stored in them differs in a functional way.

Goal of hash-consing is to optimize memory usage.

Challenges

- *Lack of distinction between hash-consed and non-hash-consed values.*

- *Difficulty in improving data structures containing hash-consed terms.*

- *Inability of the garbage collector to reclaim terms stored in the hash-consing table.*

- *Inefficiencies in the hash-consing function's implementation, involving redundant hash calculations and wasted space.*

Type-Safe Modular Hash-Consing

- *It tags hash-consed values with unique integers, creating a clear distinction between hash-consed and non-hash-consed values.*

- *Efficient data structures for hash-consed values, such as Patricia trees, balanced trees, or hash tables.*

- *Incorporates mechanisms like weak pointers to allow the garbage collector to remove hash-consed values that are no longer needed.*

- *Optimize hash key calculations, introduce efficient data structures, and offer features like total ordering over hash-consed values.*

Rust and Haskell

RUST *A language empowering everyone to build reliable and efficient software*

- *Rust is fast, provides low-level control and is memory-efficient.*
- *With no runtime or garbage collector.*
- *Has a rich type system and a unique ownership model.*

Haskell *An advanced, purely functional programming language*

- *Haskell is a general-purpose, statically-typed, purely functional language with type inference.*
- *Supports lazy evaluation.*
- *Immutable data structures.*

Motivation

- *Explore how each language's features impact the library's efficiency, memory usage, and performance.*

- *Learn more about Memory Efficiency in Functional Programming.*

- *Both languages have distinct strengths:*
  - *Rust excels in memory safety, low-level control, and performance, making it suitable for systems programming.*

  - *Haskell's functional purity and expressive type system are advantageous for mathematical and language-oriented tasks.*

Implementation in both languages allows us to leverage these strengths for different use cases.

Proposed Work

- *Develop a robust and efficient Type-Safe Modular Hash-Consing library in Rust using unique features of the language.*
- *Create an equally powerful library in Haskell using functional programming.*
- *Demonstrate how Rust and Haskell handle hash-consing in different ways.*
- *Collect extensive performance and memory usage data for benchmarking and comparison.*

Results and Evaluation

- *The libraries provide all expected features, are well-documented, and adhere to best practices.*

- *Comprehensive benchmarking to compare the memory usage and execution speed of the libraries against traditional hash-consing methods.*

- *Evaluate the success of comparative analysis between Rust and Haskell implementations.*

- *Highlight language-specific strengths and weaknesses as well as any optimization insights.*

Background

- *Jean-Christophe Filliâtre and Sylvain Conchon. 2006. Type-safe modular hash-consing. In Proceedings of the 2006 workshop on ML (ML '06). Association for Computing Machinery, New York, NY, USA, 12–19. https://doi.org/10.1145/1159876.1159880.*

- *ZHOU, N., & HAVE, C. (2012). Efficient tabling of structured data with enhanced hash-consing. Theory and Practice of Logic Programming, 12(4-5), 547-563. doi:10.1017/S1471068412000178*

- *Braibant, T., Jourdan, JH., Monniaux, D. (2013). Implementing Hash-Consed Structures in Coq. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds) Interactive Theorem Proving. ITP 2013. Lecture Notes in Computer Science, vol 7998. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-39634-2_36*

Milestones

Milestone 1: Research and Design

- *In-depth research on Type-Safe Modular Hash-Consing principles.*
- *Study existing implementations and related academic work.*
- *Formulate a detailed design plan for libraries implementation in Rust and Haskell.*

➢ *Expected Outcome: A well-defined and documented implementation plan.*

Milestone 2: Library Development

- *Implement the core features of TSMHC in both languages.*
- *Ensure type safety, modularity, and efficient memory management.*

➢ *Expected Outcome: Fully functional libraries in Rust and Haskell.*

Milestone 3: Documentation, Testing, and Performance Evaluation

- *Thoroughly document the libraries.*
- *Implement comprehensive testing strategies.*
- *Conduct performance benchmarking against existing solutions.*
- *Gather and analyze performance metrics.*

➢ *Expected Outcome: Well-documented libraries with optimized performance.*

Thank you