Optimizing lazy functional languages: utilizing reference counting for memory reuse

Martin Fredin

June 2023

1 Introduction

All values in purely functional languages are immutable. Immutability is important because it limits shared state, and makes it easier to reason about the program. In practice, this means that a function such as map returns a new list instead of modifying the input.

$$\begin{aligned}&\mathsf{map} : (A \to B) \to \mathsf{List}\ A \to \mathsf{List}\ B\\ &\mathsf{map}\ f\ [] = []\\ &\mathsf{map}\ f\ (x :: xs) = f\ x :: \mathsf{map}\ f\ xs\end{aligned}$$

This is great if the input list ($\mathsf{List}\ A$) is used later in the program. Otherwise, it is more efficient to reuse the allocation for $\mathsf{List}\ A$ by updating the list in place. This would avoid both the deallocation of $\mathsf{List}\ A$ and the allocation of $\mathsf{List}\ B$.

Precise reference counting has been proposed to identify when it is possible to update objects in place. Ullrich and de Moura (2021) uses reference counting with borrowed references in the Lean theorem prover. Reinking* et al. (2020) Lorenzen and Leijen (2022) and Lorenzen et al. (2023) also uses reference counting in the Koka language.

Both Lean and Koka are, however, eagerly evaluated.

2 Relevant Work (maybe part of introduction?)

- Lean and Koka: reference counting and reuse
- Swift: automatic reference counting (ARC), and plans of borrowing

- AST & Rust: safe manual memory management through proofs and the borrow checker, respectively

3 Future Work

- Lambda lifting
- Partial applications (P-tags and the apply operation)
- Different allocator? mimalloc?
- Using type information (multiplicites: 0, (1), ω)

4 References

References

Lorenzen, A., & Leijen, D. (2022). Reference counting with frame limited reuse. *Proc. ACM Program. Lang.*, *6*(ICFP). https://doi.org/10.1145/3547634

Lorenzen, A., Leijen, D., & Swierstra, W. (2023, May). *Fp^2: Fully in-place functional programming* (tech. rep. No. MSR-TR-2023-19). Microsoft. https://www.microsoft.com/en-us/research/publication/fp2-fully-in-place-functional-programming/

Reinking*, A., Xie*, N., de Moura, L., & Leijen, D. (2020, November). *Perceus: Garbage free reference counting with reuse (extended version)* (tech. rep. No. MSR-TR-2020-42) ((*) The first two authors contributed equally to this work. v4, 2021-06-07. Extended version of the PLDI'21 paper.). Microsoft. https://www.microsoft.com/en-us/research/publication/perceus-garbage-free-reference-counting-with-reuse/

Ullrich, S., & de Moura, L. (2021). Counting immutable beans: Reference counting optimized for purely functional programming. *Proceedings of the 31st Symposium on Implementation and Application of Functional Languages*. https://doi.org/10.1145/3412932.3412935