Modern Algorithms for garbage collection

Outlining modern algorithms for garbage collection on the examples of go, ocaml and python

Daniel Huber Matteo Gropp

October 10, 2023

Contents

1 Introduction **2**

2 Overview over Garbage Collection **3**

3 Comparison with other Memory Management Techniques **4**
3.1 Manual Memory Management 4
3.2 Lifetimes and Borrow Checking 4

4 Tradeoffs between Garbage Collection Algorithms **5**

5 Overview over current Garbage Collection Algorithms **6**
5.1 Go 6
5.2 Python 6
5.3 OCaml 6
5.4 Java 6

1 Introduction

Garbage collection describes the process of automatically allocating and deallocating memory a process requires. [1] This is commonly performed by the runtime the compiler embeds into the resulting executabe or the runtime the interpreter provides to the program it currently executes. [2, 3]

2 Overview over Garbage Collection

3 Comparison with other Memory Management Techniques

3.1 Manual Memory Management

3.2 Lifetimes and Borrow Checking

4 Tradeoffs between Garbage Collection Algorithms

5 Overview over current Garbage Collection Algorithms

5.1 Go

5.2 Python

5.3 OCaml

5.4 Java

References

[1] *Garbage collector design*. URL: https://devguide.python.org/internals/garbage-collector/index.html (visited on 10/09/2023).

[2] *A Guide to the Go Garbage Collector*. URL: https://tip.golang.org/doc/gc-guide (visited on 10/09/2023).

[3] *gc - Garbage Collector interface*. URL: https://docs.python.org/3.13/library/gc.html (visited on 10/09/2023).