Compacting Garbage Collection can be Fast and Simple

Charles L. A. Clarke David V. Mason
Department of Computer Science
University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
claclark@plg.uwaterloo.ca
dmason@plg.uwaterloo.ca

March 6, 1996

Abstract

Copying garbage collectors are now standard for the memory-management subsystems of functional and object-oriented programming languages. Compacting garbage collection has correspondingly fallen out of favor. We revitalize the case for compaction by demonstrating that a simple compacting collector, extended with the generational garbage collection heuristic, exhibits performance as good as or better than a well-designed generational copying collector on real programs running in real environments. The observation that compaction preserves allocation order across collections leads to a new generalization of the generational heuristic that reduces the movement of long-lived objects. We measure the effect of substituting our compacting generational collector for a copying collector in Standard ML of New Jersey.

This note is placed at the WWW URL:

`http://plg.uwaterloo.ca/~claclark/gc.ps`

to replace the technical report that previously appeared there.

A version of the technical report now appears in print [1], and the technical report has been withdrawn. Unfortunately, due to a publishing error, the journal version of the paper has an incorrect figure 1. The correct figure is included in this note. If anyone can identify the source (or explain the meaning) of the figure that appears in print, we'd be happy to hear about it.

Thanks for your interest.

References

[1] Charles L. A. Clarke and David V. Mason. Compacting garbage collection can be fast and simple. *Software — Practice and Experience*, 26(2):177–194, February 1996.

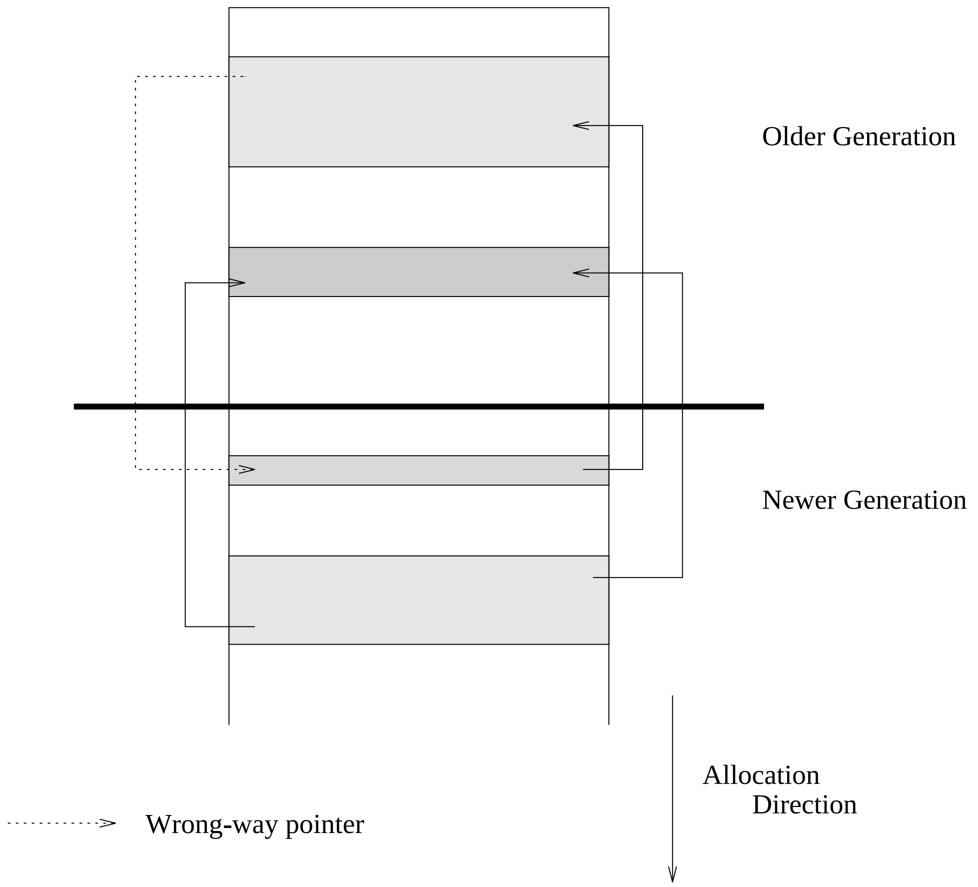

Figure 1: Generational garbage collection