Technical Articles: A Special Issue on "Garbage"

In this issue of Lisp Pointers, we are featuring two articles on memory management for Lisp systems: *Overview of Garbage Collection in Symbolic Computing* by Timothy McEntee is an excellent short tutorial on the current state of garbage collecting techniques; and *Address/Memory Management for a Gigantic LISP Environment, or GC Considered Harmful* by JonL White proposes a novel technique for use in large virtual memories that abandons the notion of "collecting" garbage and focuses instead on the problem of maintaining a relatively small working set. Both of these articles have appeared elsewhere, in forums with very limited distribution, and so are being reprinted here. The first article originally appeared in longer form in the **Texas Instruments Engineering Journal**, which is primarily distributed to TI employees; the second article originally appeared in the historic 1980 Lisp Conference. [Since 1982, the biennial Lisp Conferences have enjoyed ACM sponsorship, and the proceedings are more widely distributed.]

The Mark-and-Sweep techniques of garbage collection have apparently fallen into into disfavor, probably due to the increased importance in Lisp of variable-length objects such as arrays and defstructs [cons cells, symbols, floating-point numbers, etc., are typical fixed-length objects]. Only the Interlisp-D workstations marketed by Xerox AI Systems rely mainly on such schemes. However, some new implementations, such as ZIL [Lisp Pointers, vol 1, no 2], as well as the classic PDP10 MacLisp, profitably combined Mark-and-Sweek algorithms with Stop-and-Copy ones. But the prominent problem in Lisp memory management, since the mid 1970s, is one of working set size. McEntee indicts the classical algorithms as having the shortcoming of "severe thrashing," and even Baker in an MIT AI Lab memo criticizing his *incremental* collector scheme noted that merely compacting the total data set did not necessarily reduce the size of the working set (in fact, it tended to do just the opposite!).

A word on names for memory management algorithms: the terms Mark-and-Sweep and Stop-and-Copy are well-entrenched, and the class of prominent variants on Mark-and-Sweep called Reference Counting is also well-established as a name. The name Baker's Algorithm has generally been used to refer to Henry Baker's variant of Stop-and-Copy [but the nearly real-time nature of it suggests that it be named the Dont-Stop-but-Copy algorithm]. The variant proposed in the second paper has been referred to by its author as the Dont-Do-It scheme of garbage collection, although others, noting the desirability of the goal in section 3, have termed it Annual-Garbage-Collection. Finally, two terms have come into common use for the variant of Stop-and-Copy proposed by Hewitt and Lieberman: Generation-Scavenging and Ephemeral, the latter term coming from Moon's paper describing Symbolics' implementation of generation scavenging on its 3600 processor. Perhaps *Ephemeral* should denote the mixture of the incremental nature of Baker's algorithm with the partitioning of the heap into generations, whereas unqualified *Generation Scavenging* would refer to all such schemes, including implementations like the Berkeley Smalltalk where frequent mini-pauses occur during which all processing stops except for the "scavenging" of some or all of the "generations" (estimates for a time bound on these mini-pauses in other similar systems range from a couple hundred milliseconds to six seconds).

JonL White

Technical Articles Editor