



backtracking

Still Garbage Collecting

Way back in the good old days — when men were men, women were women, and hackers were indeterminate — if you were doing AI, you were doing LISP. LISP is a language that takes years to appreciate, like a taste for wine, a Buddhist koan, or a leaf on a tree. It is infinitely deep, with hidden elegance and secrets that cannot be understood without serious contemplation.

LISP was never for the meek, never for the vulgar, never for the masses, never for the *Web*.

Back in those days, AI practitioners who wished to practice their art in industry — perhaps to gain more users for “usage data” or to make more money to buy some more Mountain Dew — had to demonstrate an application of their ideas. Many of us have had to give these “demos.”

The inevitable question asked during a demo was always, “What language is it written in?” This question was useful because it could be asked at the end of an hour-long nap and still seem relevant to the discussion and presentation that occurred during the nap. In addition, all work could be evaluated quite simply based on the answer to this question, as the table below (taken from “A Manager’s Guide to Dealing with Researchers”) shows:

PERMISSION TO MAKE DIGITAL OR HARD COPIES OF ALL OR PART OF THIS WORK FOR PERSONAL OR CLASSROOM USE IS GRANTED WITHOUT FEE PROVIDED THAT COPIES ARE NOT MADE OR DISTRIBUTED FOR PROFIT OR COMMERCIAL ADVANTAGE AND THAT COPIES BEAR THIS NOTICE AND THE FULL CITATION ON THE FIRST PAGE. TO COPY OTHERWISE, TO REPUBLISH, TO POST ON SERVERS OR TO REDISTRIBUTE TO LISTS, REQUIRES PRIOR SPECIFIC PERMISSION AND/OR A FEE. © ACM 1523-8822 99/0900 \$5.00

ANSWER	RESPONSE
C	C. That’s good. Very fast and efficient.
Pascal	<raise eyebrows> A toy language, but it’s easy to parse, which is good.
PL/1	<chuckle> Do they still use that?
FORTRAN	<show concern> FORTRAN is really only for math, because it can’t handle recursion.
APL	No, I meant what <i>programming</i> language.
LISP	<smile as if talking to a naive child> LISP isn’t for real software because it is interpreted, and it does <i>garbage collection</i> . In the real world, you see, we do not have time to wait while a computer collects garbage.
ProLog	<see LISP>

The real pitfall of LISP systems was that they actually displayed something on the screen when they were garbage collecting. Managers were perfectly willing to wait for a C program to finish some mysterious calculation (“What’s it doing now?” “Finding the root of a large polynomial using Newton-Raphson.” “Ah, excellent”), but completely unwilling to wait a few seconds staring at the letters “GC.”

One of the earliest contributions of the Web was that LISP programs could be written to run on a Web server, with all UI handled by an http connection from a Web browser. The advantage here was that when the LISP system went into GC, it would suspend communication on the http stream, and the Web browser would invariably display, “Host contacted, waiting for reply.” This was infinitely more palatable than “GC.” At this point the demo-giver could say, “Netscape is hung for some reason” or “The network is sure slow today,” knowing full well that over on the server the letters “GC” were flashing.

Then came Java. At first, many of us old-time LISP hackers were offended by Java — it does nothing that LISP can’t do, so why bother with it? (In fact, this is true of most things.) However, Java does things the right way:

- ◆ First of all, Java is *object-oriented*, which is good. LISP is functional, which is something else.
- ◆ Second, Java comes with extensive libraries of functions. LISP applications use too much memory.
- ◆ Third, Java *comes with its own virtual machine*. LISP is interpreted.
- ◆ Fourth, Java is portable. LISP only runs on machines with a lot of memory.
- ◆ Finally, Java *does its own memory management*. LISP uses garbage collection.

Apparently, what Java has is spin doctors.

— Chris Welty and Lou Hoebel,
editors@sigart.acm.org