QuadAV Considered Harmful – Still*

Robert Bernecky

Snake Island Research, Inc.
18 Fifth St., Ward's Island, Toronto, Canada M5J 2B9
tel: 416 203 0854
email: bernecky@acm.org

September 1990

Abstract

The APL character set has contributed, more than any other facet of the language, to its lack of acceptance in the computing community at large. The character set is a metaproblem – not a problem in and of itself, but a creator of other recurring problems of hardware, software, ergonomics, and psychology. The adoption of new, ASCII-base dialects of APL, such as J, is suggested as one solution to the character set problem.

Ergonomics Joke

This guy walks into the Ye Olde APL Tailor Shoppe, and wanting to buy a suit. The tailor pulls one off the rack, and has him try it on. The guy says to the tailor that the jacket doesn't fit right – one arm is much longer than the other.
"No problem", says the tailor, "Just hunch up your shoulder on that side, and it'll look fine."
"The lapels are uneven." says the guy.
"Oh, that. Well, push it over with your chin."
The guy also notices that the pants aren't the same length. "Hitch up your hip on the longer side, and it'll be fine. Now, my! Doesn't that look great!"
The guy gimps out of the tailor shop wearing his new suit, and passes two old ladies walking the other way. After he passes, the one says to the other:
"Oh, that poor old man. Did you see how crippled up he was?"
"Yes, but doesn't his suit fit him well?"

1 Introduction

APL's character set sets the language apart from all other computing languages. Although the character set provides a terse notation which enhances our thought processes [Be90a, Be90b, Iv79], it also presents a number of problems, both technological and psychological. the problems engendered by the character set outweigh its benefits. The success of APL-like languages will only be achieved by

*Originally published in ACM SIGAPL *Quote Quad*, vol.21, no. 1, September 1990.

abandoning the APL character set, and adopting an ASCII-based notation.

2 Why ⎕av is Swell

Let's begin by examining the claims made for the APL character set, and the arguments for its use:

- APL symbols are visually compelling
- Display of APL is no longer a technical problem

There is no doubt that APL symbols are visually compelling, and that they offer an experienced user a leg up in comprehension of programs. Idioms such as `×/⍴⍵` and `((⍳⍴⍵)=⍵⍳⍵)/⍵` have clear and obvious meanings as the element count in an array, and the set of unique elements in a list.
Throughout the history of APL, the character set has been a technological problem area: Special keyboards, displays, print trains, ROMs, and other devices were required in order to write or display APL programs. These problems were solved over time, over and over again, for each new device or system for which APL support was desired.
With bit-mapped technologies, we are able to display garbage cans and other such icons, and the claim is made that, with this sort of display capability, APL support is no longer a hardware problem. Furthermore, many common editors, such as EMACS and PCWRITE, and IBM mainframe editors, have been tinkered by APL fen[1] to support APL characters. Therefore, software is no longer a problem area.

[1] *fen*: plural of fan, from science fiction fan magazines, although in this context, perhaps fanatic is a better etymology.

Or is it?

3 Hardware Problems

Let's start by examining the claim that the hardware problems have been solved, and tie that claim into the known problems of getting APL into a new shop and the ergonomic issues of training new users.
It is clear that laser printers and modern video displays can be made to display arbitrary bit patterns, including APL characters. However, there is no key on a keyboard which is defined to have a garbage can, or a transpose character on it – this work is done by an application program, and is not part of the basic box as delivered. There is an up-front cost associated with support for APL, *even if* the underlying machine is able to support it.
Display of APL characters requires provision of ROM or programmable fonts for the particular display being used.
High-quality impact printers require special, expensive APL print trains. Transmission controllers for IBM 327X devices require special hardware which is often incompatible with other options, and which also adds extra costs. Support for APL on LAN-connected printers is a maintenance nightmare.
Finally, and most important: the costs are NOT one-time costs. They are recurrent costs, which crop up again and again, each time new software or hardware is installed in the shop. Consider something as simple as APL keycaps for a terminal. Multiply $95 by the number of 327X devices in a shop, and again each time a new flavor of 327X supplants the old one, and it becomes clear that recurring costs become quite high, particularly for a capability which is only

required for a single computing language. Furthermore, if you are using more than one APL system, the keycaps or sticky labels for one may not match the other(s). Do you then go out and buy multiple keyboards, plugging them in as you need them?

4 Software Problems

Support for laser printers requires that someone define a character set mapping (which transmission code means which APL character), create a font for the character set, create procedures for downloading or otherwise getting the font to the printer, modify editors (if you have access to the source code) to allow entry of APL characters, and so on.

Software costs associated with APL character set support are often hidden and insidious. Every time a new piece of hardware is installed in the shop, device drivers and interface code has to be modified to support the APL character set. When a new release of an operating system is installed, it may contain changes which make support of APL difficult – a previously modified editor may have to be modified again. Perhaps the changes are extensive enough that modifications have become impractical. When I left I.P. Sharp Associates to found Snake Island Research, I bought a shiny new laser printer. It took me more than a month of effort, and a thousand dollars in extra hardware, to get it to the point where I could print APL. How long would it have taken someone who was just starting to use APL?

What's to be done when an editor changes? Use the old editor? Keep both of them around, creating a maintenance nightmare for system programmers? Design, write, and install a custom editor, which creates maintenance, portability and training problems for all involved?

PCWRITE, a shareware editor for PCs, was modified to allow entry of APL characters, but it is very easy to get into trouble with the modified version. Data sensitivities in other pieces of system software create subtle, insidious problems. For example, if you decide to display a PC file using the TYPE command, the display truncates at the first APL right arrow, because the arrow is stored as an end-of-text character.

5 Ergonomics

Entry of APL characters requires modification of the keyboard mapping. It may also require provision of keycaps or sticky labels to help you navigate around the keyboard. These modifications may be incompatible with windowing systems or other software, including other APLs, in place on the machine. If a non-union approach to keyboard mapping is taken, then there is a high cost of training new users to the concept that the parenthesis marked on the keyboard is not the *real* parenthesis, and that the *real* parenthesis is located on the key where the quote is. The quote, of course, is on the key labelled K, and...

The next time you try to introduce someone to APL, note how easy it is to describe the concepts of verbs, adverbs, and array operations on a blackboard. Then, sit them down at an APL keyboard, and watch how quickly they become frustrated with the interface.

The historical inversion of APL alphabets makes things even worse – if you hit an

unshifted r, it displays as R. If you type it shifted, it appears as ←, and if you use CTRL r, you get either r or an underbarred R. Great. Now, teach this to a trained typist, and let me know what sort of response you get.

There are ways around all these problems, but each workaround is unique, subject to change, and a barrier to ease of use by people who are already familiar with other modes of use of a system.

6 Psychological Problems

There was and is a psychological barrier to use of APL, caused by its character set – "What are those funny squiggles?" This barrier very often stops people from taking the time to learn the merits of APL. They see the characters as being foreign, and a xenophobic reaction sets in, which prevents them from being open-minded enough to look closer.

The availability of garbage cans on screens may alleviate some of this problem as people get accustomed to "icons", but it is doubtful if the barrier will ever disappear completely.

APLers spend a lot of time trying to convince neophytes of the virtue of the APL character set, and we often lose this battle, leaving the neophyte to the world of Fortran, BASIC, or PASCAL. Why do we continue to carry on the character set crusade? Is the character set that much an important part of APL? Is ← more important than the concept of adverbs? No. The crucially important aspects of APL are its power, simplicity, and consistency!

7 Fear, Uncertainty, and Doubt

Consider the answers to the following questions, assuming you are an APL programmer:
Will a new editor or windowing system support my characters, and allow their display? Will my new Cray support the symbol set? Can I transfer programs from one system to another? Can I print programs on my new printer? Can I use a new editor, that only supports a 128-character ASCII alphabet? Will it work without modifications? Can I tell how long will it take to modify it? Can I upload or download programs to my PC?
The answers to each of these questions vary for APL, depending on the specific nature of each facility and system. These are APL problems which are usually solvable in a given system, given enough time, skilled personnel, and resources. However, they must be solved over and over again, with every change in environment – once is not enough.
Now, think about programs written in ASCII-based languages. How many of the above problems exist for them? None. What is the direct and obvious answer to all the questions above? Yes.
The fact that other languages work within an existing framework means that the time to utilize standard system facilities is zero. Not small, not some, not unknown – zero.
Is it any wonder that it's hard to get computer shops to install, run, and maintain APL?

8 Publication Problems

Have you ever seen APL programs or expressions turned into gibberish by publishers?

Probably almost every one which was not published by Quote Quad, is my bet. Even magazines which should be aware of such problems seem unable to publish even the simplest APL expressions without error. My letter in the September 1990 issue of *Computer Language* contains an APL expression of about ten characters, in which the iota has fallen over, and become a tilde.
I can send publishers complete TeX or LaTeX files, and have them printed with no problems, unless they contain APL characters, in which case, all bets are off. By avoiding the APL character set, we avoid having to render fonts and assistance to all publishers on the planet.

9 Marketing and Sales Problems

The problems of selling and marketing APL are demonstrated by the following scenario, which I saw happen repeatedly during my tenure with I.P. Sharp Associates:
Prospective Client: Can you send me a demo APL program to try out?
Hapless Sales Type: SURE! No problem! Just call me when you have stickers or keycaps on all the keyboards, have all the TP controllers upgraded and reconfigured, and have the special-ordered APL print train available, OK?
Prospective Client: (Sound of phone being hung up.)
APL's character set requires an infrastructure which represents a significant investment in human resources, hardware, and software, before *any* benefit can be shown. This is a killer when trying to convince a potential client of the benefits of APL. It's like trying to sell someone a car, and telling them that they'll have to acquire land and build a freeway before you can demonstrate its virtues.

10 Metaproblems

At the 1990 Minnowbrook Implementors' Workshop, hosted by Garth Foster and Syracuse University, I presented a session which turned into this paper. Cory Skutt made the observation that the APL character set is a *metaproblem* – it is not, by itself, a problem. However, it *creates* problems, on an ongoing basis!
We have seen that the APL character set creates problems which affect operating system code, displays, editors, terminals, printers, controllers, human interaction, and so on. These problems must be solved at three points in time:

- Before APL is ever even used at the site
- Whenever hardware changes occur at the site
- Whenever software changes occur at the site

The question boils down to a tradeoff between the benefits provided by the notation, and the problems the notation creates. Is the APL character set worth it?
I claim the answer is an emphatic *NO*, if we ever want to integrate APL into the rest of the computing world. The only solution is to fit into the environment by adopting the environment's existing mechanisms and tools, without change. This means that we have to abandon the APL character set, and adopt ASCII.

11 ASCII and the Metaproblem

The adoption of ASCII severs the Gordian Knot which has strangled APL's growth. Hui and Iverson have done this with J,[Hu90], allowing them to:

- Run on any system with no preparation
- Use any printer
- Use any publisher
- Use any text editor

The readability of an ASCII-based notation need not be substantially less than one based on `av`. Here are two examples from J, using reshape (`$`), the integers generator (`i.`), and replicate (`#`):

```
      2 4$ i.  8
0 1 2 3
4 5 6 7

      1 0 1 1 1# 'board'
bard
```

If the APL community continues to remain an insular, inward-looking body, APL will wither and die, much as Algol has done. If, however, we wish to have our ideas accepted at this time when parallel thinking is finally becoming fashionable, we must speak the language of the natives who we wish to convert. Those natives speak ASCII.

In closing, note that the use of ASCII as a notation for APL will have two benefits above and beyond those described above:

- It will mean that we can stop crusading to get our ideas and concepts across to the world, and go back to solving problems with APL.
- It will put an end to jokes about APL.

References

[Be90a] Bernecky, R., *Fortran 90 Arrays*, in press.

[Be90b] Bernecky, R., *Ergonomics and Language Design*, to appear.

[Hu90] Hui, R., K.E. Iverson, E.E. McDonnell, and A.T. Whitney, "APL\?", *APL90 Conference Proceedings, APL Quote-Quad*, Vol. 20, No. 4, July 1990.

[Iv79] Iverson, K.E., "Notation as a Tool of Thought", *Communications of the ACM*, 1979 Turing Award Lecture, 1979.