Letters

If versus Rule

Dear Editor,

I am writing about the article "The Time is Ripe for a Dyadic Execute" by Zdenek V. Jizba that appeared in APL Quote Quad Vol. 19, no. 2.

I would like to comment on Jizba's characterization of a dyadic execute as being akin to a "rule" in expert systems languages.

His dyadic execute usage is more accurately described as an if/then operation, not a "rule".

A rule, in most AI and expert systems usages, is an operator that has, in its simplest form, two parts: a left hand side and right hand side, also know as patterns and actions.

IF patterns THEN actions

Here the similarity with a procedural language's IF/THEN operation stops. A rule does not need to be EXECUTED to cause the right hand side (actions) to "fire". Rules are usually generalized demons, that is, they are constantly looking at the data, and when they see a pattern in the data that matches their left hand side, they can execute the actions on their right hand side.

Once a rule is executed (also called "compiled," or "added to the rule set"), this "lookout for the pattern" action is always active. The test, or the pattern on the left hand side, is not done only when the rule is executed, like an IF/THEN statement. It is always active, ready to jump when it finds some data that matches its left hand side.

For example, in a procedural language, an IF/THEN does its testing only when the IF/THEN statement is executed. (Assume "=" is assignment.)

```
    x = 7
    IF x > 8 THEN print "hello"
    x = 9
  Results: nothing
```

The IF statement would have to be "executed" again to get it to recognize that x has indeed taken a value greater than 8.

But in a rule, the testing continues after the rule has been executed (or compiled, or put into the rule set).

```
    x = 7
    IF x > 8 THEN print "hello"
  Results: nothing
    x = 9
  Results: prints "hello"
```

So, while the recommendation that dyadic execute should be added to the APL language is worthy of consideration, please don't call this function "rule." It would be useful as a simple IF/THEN function.

APL has been used successfully on many Artificial Intelligence and Expert Systems applications already, with or without a built-in IF/THEN function.

Sincerely,

Beth Tibbitts
Room 35-210 (914) 945-2411
IBM TJ Watson Res. Ctr.
P.O. Box 218
Yorktown Heights, NY 10598
Bitnet address: BRT at YKTVMZ, BRT at WATSON,
or brt @ ibm.com

To the Editor:

Three comments on the definition of the dyad ⍎ proposed by Z.V. Jizba in APL Quote Quad, 19 no. 2.

1. Since the explicit result of *L⍎R* is simply *L*, useful results can only be obtained as side-effects in *R*.
2. Conditional execution of a function can be obtained as a special case of the "power conjunction" (denoted by .) defined on page 34 of Quote Quad, 18 no. 1.
3. The alternative definition of the dyad ⍎ offered in Quote Quad, 18 no. 1 provides a form of "trap" or "exception handling" not otherwise available in the language.

Kenneth E. Iverson
70 Erskine Ave.
Apt. 405
Toronto, Ontario
M4P 1Y2

Dear Editor,

As the countdown to APL89 continues, I would like to extend a challenge to various implementers of APL. Please help me to express the functions in *Probability in APL* in your favorite dialect. I, and most educators, are in the difficult position of having to choose between a variety of expressions for the same concepts. Please submit all entries to me before the end of the Conference to become eligible for an APL Medal.

Linda Alvord
Scotch Plains-Fanwood High School
Scotch Plains, NJ 07076
USA

To the Editor,

'Fast I/O - Efficient File Processing'

The above paper (by Lori D McNichols) was distributed at the APL88 conference and gives a thorough exposition with examples of a very useful technique - I would hope that it has wider dissemination because these are real performance improvements that anyone working in the APL2/TSO environment can implement for very little cost and effort. Use of this code in recent months has led to two observations which I would like to share.

a) As presented the function <CHECK_DCB> is unnecessarily restrictive, modification to lines [7] and [8] permits use of the functions on partitioned datasets and those with RECFM=U.

b) A circumstance has arisen whereby the performance gains are not achievable; namely when a dataset is being extended using DISP=MOD. In which case the following happens:

   ISPF (etc.) reports BLKSIZE=bignumber

   AP111 finds that the blocks are nowhere near this long - in fact it retrieves chunks which are as long as the bits that got added - which is especially bad news if you have an application which adds records one at a time.

Clearly we have a useful technique here, and one more widely applicable than the original exposition suggested; nevertheless there is also a lesson to be learnt in that we have to stay vigilant and measure everything - because even gift horses are not always what they seem.

Dick Bowman
c/o CEGB
85 Park Street
London SE1
England

Correction

The previous issue of Quote Quad (19 #2) had a few lines which could hamper easy understanding.

The last 10 lines of paragraph 4 of page 18 should read instead:

```
  Y[(,~I)/ιρ,I]←0
```

instead of the better

```
  I[(,~I←X≠MISSV)/ιρ,I]←0
```

and

```
   ... ⋄ I←0×N←ρD ⋄ ...
  L1:    ...
         ...
         →(N<I←I+1)/L1
```

instead of the better

```
   I←0
   L1: ...
       ...
       →((ρD)<I←I+1)/L1
```

F.H.D. van Batenberg

Mid-line assignment

Dear Editor,

Your editor's note in APL Quote Quad, 19/2, p. 19, would have been even closer to the point if the following code were included:

```
[2]    ...
[3]    ...
[4]    Y[(~I)/ιρI←,X≠MISSV]←0
```

after removing the first ← from the article, too.

Adam Kertesz
400 East 58th Street
New York, NY 10022
USA

[Editor's note: One who moves APL code from one APL implementation to another becomes aware of trouble spots that exist because not all implementations agree about how lines of code should be paired. One such problem area is mid-line assignment.

Readers are invited to submit examples of code they have found that does not port from one APL to another. In preparing the example, please trim the code to the bare essentials that illustrate the point.]

Dear APL Colleague,

The Toronto APL Special Interest Group is pleased to announce their new publication, The APL Toolkit, 2nd Edition, available immediately. It is the result of our group effort over 5 years!

For mailing addresses in Canada, the price of the Toolkit manual is $25 Canadian. For all other locations, the price is $25 U.S. This is an exceptional bargain, considering that it costs almost as much to produce and mail. If you wish to order, please remit the appropriate amount by cheque or money order to the following address:

CIPS APL SIG - Toronto
Attention: Toolkit Editor
P.O. Box 384, Adelaide St. Station
Toronto, Ontario, Canada
M5C 2J5

We are sure you will agree that this is one of the finest collections of APL functions for the general audience. It is fully documented, and you will learn much about APL pro-

Continued on page 27