CS 421 Lecture 15

- Today's class: APL
  - Functional programming – "no side effects"

$\text{if } (f() + 1 > f()) \ldots$

Functional Programming

$\cancel{e + 1 > e}$

- "The assignment statement splits programming into two worlds. The first world comprises the right sides of assignment statements. This is an orderly world of expressions, a world that has useful algebraic properties.... It is the world in which most useful computation takes place.
- "The second world... is the world of statements. … This world of statements is a disorderly one, with few useful mathematical properties."

John Backus (creator of Fortran), "Can Programming be liberated from the von Neumann Style? A Functional Style and its Algebra of Programs." Turing Award lecture, 1977.

$\cancel{e_1 + e_2 = e_2 + e_1}$

APL

- Computations on matrices using operators that have matrix arguments.
- Ken Iverson – "A Programming Language" – 1960
- Defined a set of operators on matrices, plus a typeface for those operators, and built terminals

APL operations

- Binary operations on numbers extended naturally to matrices
  - Comparison and boolean ops treated as arithmetic
- Reduction operations: $+/$, $\times/$, $\wedge/$, …
  - For vectors, put operator between every element
  - For matrices, reduce each row
- Compression: B / V
  - selects elements (or rows) of V where B = 1

boolean (0's & 1's)

No precedence rules, evaluate right-to-left

Monadic - unary

Dyadic - binary

APL font

ABCDEFGHIJKLMNOPQRSTUVWXYZ

0 1 2 3 4 5 6 7 8 9

index gen

and

assignment

≤ ≥ ≠ < > = ⍳ ⍴ ⍉ ∧ / ÷ ← ⋆ × + − arith op's

comparison

transpose

dyadic: compression also used in reduction op's

monadic: shape (dimensions)

dyadic: restructure

APL examples

- $1 + M$ — $\begin{pmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \end{pmatrix}$
- $(+/V) \div n$ If $n = 4$, gives average — 5
- $(+/V) \div \rho V$ — Average of V

  (+/M) ÷ ρM

  6 15 ÷ 2 3

  = 3 5

  would not work if M had >2 rows
- $(((V \div 2) \times 2) = V) / V$

$V \leftarrow 2\ 4\ 6\ 8$

$M \leftarrow 2\ 3\ \rho\ 1\ 2\ 3\ 4\ 5\ 6 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

APL examples

- `prime n = ∧/(0≠n%(1+ι(n-2)))`

$n*.5$

x

1 2 3 ... n-2

2 3 4 ... n-1

n%2 n%3 ... n%n-1

0's where zero here,
1's where non-zero

APL examples

- Subscripting: $V[V']$ – elements of V in positions given by V'.

Suppose $\rho V = n$

reverse V = $V[1+(\rho V)-\iota\rho V]$

$n \quad - \quad 1 \ldots n$

$n-1 \quad n-2 \quad \ldots \quad 0$

$n \quad n-1 \quad \ldots \quad 1$

$V[n] \quad V[n-1] \quad \ldots \quad V[1]$

$V = 2\ 4\ 6\ 8$

$V[1\ 2\ 1\ 2\ 1\ 2] = 2\ 4\ 2\ 4\ 2\ 4$

APL examples

- Dyadic ρ – "restructure"
- $V \rho A$ returns a value with shape V, values drawn from A

$$2\ 3\ \rho\ \iota 6 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

$$2\ 3\ \rho\ \iota 5 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 1 \end{pmatrix}$$

$$\underbrace{(2 \rho n)}_{n\ n}\ \boxed{\rho}\ 1,\ \underbrace{n \rho 0}_{\text{vector of } n\ 0\text{'s}} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 \cdots & 0 \\ 0 & 0 & 1 & \cdots \\ & & \ddots & \\ & & & 1 \end{pmatrix}$$

$$1\ \underbrace{0 \cdots 0}_{n\ 0\text{'s}}$$

APL examples

quote quad

$⍞ \leftarrow$

- $\leftarrow$ assignment
- ⍉ transpose

$$I \leftarrow (⍉M) \boxed{=} M \leftarrow \underbrace{(2\,\rho\, n)}_{n\ n}\,\boxed{\rho}\,\underbrace{\iota\, n}_{1\ 2\ \cdots\ n}$$

$$\begin{pmatrix} 1 & \cdots & 1 \\ 2 & \cdots & 2 \\ \vdots & & \\ n & \cdots & n \end{pmatrix} \qquad \underbrace{\begin{pmatrix} 1 & 2 & \cdots & n \\ 1 & 2 & \cdots & n \\ \vdots & & & \\ 1 & 2 & \cdots & n \end{pmatrix}}_{M} \Big\} \; n \times n$$

APL examples

```
let zero = newint 0;;
let four = newint 4;;
let a = rho(newveci [2;3]) (indx (newint 6));;
let v = newveci [2;4;6];;
let c = newveci [1;0];;
let d = newveci [1;0;1];;
a *@ a
v -@ one
a >@ four
!+v
```

2 3 ρ ι 6

a × a

v − 1

a > 4

arith. op's

+-reduction

APL examples

maxR a , minR – max reduction
min reduction

d % v
c % a } compression

shape a

ravel a — turns matrix or scalar into vector

rho (shape a) v

rho (shape v) c

a ^@ c – catenate

trans – transpose

@@ subscript

shar

APL examples

indx (newint 5) ⍳5

trans a ⍉a

v @@ (indx two) v[⍳2]

a @@ one a[1]

(trans a) @@ (indx two) (⍉a)[⍳2]

APL examples

```
let incr a = a +@ (newint 1);;
let fac n = !* (indx n);;
let avg v = (!+v) /@ (shape v);;
let reverse v =
  let sz = (shape v) @@ one
  in v @@ (incr (sz -@ (indx sz)));;
let prime n = !& (zero <>@ (n %@ (incr
  (indx (n -@ two)))));;
```

APL

a + 1

/* ⍳n

(+/ v) / ⍴v

let sz = (⍴v)[1]

in v[1+ sz - ⍳sz]

- Same as prime in APL