Boolean Scans and Reductions

Richard Park

Booleans in APL

Logical values are the numbers 0 and 1

Comparisons

```
      _ ← ,⍤⍳⊢    ⍝ Stranding function

      ¯2 ¯1 0 1 2 (↑ < _ ≤ _ = _ ≠ _ ≥ _ > ) 0
1  1  0  0  0
1  1  1  0  0
0  0  1  0  0
1  1  0  1  1
0  0  1  1  1
0  0  0  1  1
```

Data-driven conditionals

```
      ↑ groups salaries
    A     B     A     C     C
20000 25750 21000 32350 32400

      salaries × 1.05 * groups = 'A'
21000 25750 22050 32350 32400
```

Selecting from arrays

```
      ↑ values names
```

4	10	6	8	16	24
Anne	Ben	Charlie	Demi	Ella	Fiona

```
      (values≥10)/names
```

Ben	Ella	Fiona

Simple statistics

```
      (+⌿÷≢)'the alphabet'∊'aeiou'
0.3333333333
```

Sixteen logical functions

Phil Last

Boolean Reductions

archive.vector.org.uk/art10501140

Sixteen logical functions

0f0	0f1	1f0	1f1	2⊥	f	
0	0	0	0	0	0⍨	false
0	0	0	1	1	∧	and
0	0	1	0	2	>	left but not right
0	0	1	1	3	⊣	left
0	1	0	0	4	<	right but not left
0	1	0	1	5	⊢	right
0	1	1	0	6	≠	exclusive or
0	1	1	1	7	∨	or
1	0	0	0	8	⍱	nor
1	0	0	1	9	=	exclusive nor
1	0	1	0	10	~⍥⊢	not right
1	0	1	1	11	≥	left or not right
1	1	0	0	12	~⍥⊣	not left
1	1	0	1	13	≤	right or not left
1	1	1	0	14	⍲	nand
1	1	1	1	15	1⍨	true

0	f	0
0	f	1
1	f	0
1	f	1

Scans and Reductions

Reductions: Summarise

Scans: Indicate progression

Two well known

```
      ∧/1 1 1 1 1 1 1 1   ⍝ Are all true?
1
      ∧\1 1 0 1 1 1 0 1   ⍝ Were all true so far?
1 1 0 0 0 0 0 0
```

Two well known

```
      ∨/0 0 1 0 0 0 0 0   ⍝ Are any true?
1
      ∨\0 0 1 0 0 0 1 0   ⍝ Were any true so far?
0 0 1 1 1 1 1 1
```

The last value

```
      ∨/0 0 1 0 0 0 0 0   ⍝ Are any true?
1
      ∨\0 0 1 0 0 0 1 0   ⍝ Were any true so far?
0 0 1 1 1 1 1 1
```

The last value

```
      ∨/0 0 1 0 0 0 0 0   ⍝ Are any true?
1
      ∨\0 0 1 0 0 0 1 0   ⍝ Were any true so far?
0 0 1 1 1 1 1 1
```

The last value

```
      ∨/0 0 1 0 0 0 0 0   ⍝ Are any true?
1
      ∨\0 0 1 0 0 0 1 0   ⍝ Were any true so far?
0 0 1 1 1 1 1 1
```

Less-than

```
        <\0 0 1 0 1 1 1 1 1 1 0 1 0 1 1
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
        </0 0 0 1
1
```

Less-than

```
      <\0 1 0 1 1
        0           →  0
        0<1         →  1
        0<1<0       →  0
        0<1<0<1     →  0
        0<1<0<1<1   →  0
```

Less-than

```
      code ← '+/⍳10   ⍝ sum⍝of⍝first⍝10⍝ints'
      c ← code = '⍝'
      ↑ code c (<\c)
+ / ⍳ 1 0       ⍝   s u m ⍝ o f ⍝ f i r s t ⍝ 1 0 ⍝ i n t s
0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
      c⍳1
9
```

Less-than

```
      v ← '⊛□⊛⊛⊛□□⊛□□⊛⊛⊛⊛□□□□⊛⊛□□□□⊛⊛□⊛'
      q ← '□'=v
      ↑v q (2</0,q)
⊛ □ ⊛ ⊛ ⊛ □ □ ⊛ □ □ ⊛ ⊛ ⊛ ⊛ □ □ □ □ ⊛ ⊛ □ □ □ □ ⊛ ⊛ □ ⊛
0 1 0 0 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0
0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0
```

Not-equal

```
      ≠/1 1 1 1
0
      ≠/1 1 1 0
1
      b ← 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1
      ↑ b (≠\b)
0 0 1 0 0 0 1 0 0 0 0 1 0 0 1
0 0 1 1 1 1 0 0 0 0 0 1 1 1 0
```

```
      2|+/1 1 1 1
0
      2|+/1 1 1 0
1
```

Not-equal

```
      ≠\0 1 0 0 1
        0           →  0
        0≠1         →  1
        0≠1≠0       →  1
        0≠1≠0≠0     →  1
        0≠1≠0≠0≠1   →  0
```

Not-equal

```
      quoted ← 'extract the "quoted" parts from this "text"'
      '"'=quoted
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
      (~∧≠\)'"'=quoted
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0
      (⊢∨≠\)'"'=quoted
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
```

Not-equal

```
      quoted ← 'extract the "quoted" parts from this "text"'
      '"'=quoted
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
      (~∧≠\)'"'=quoted
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0
      (⊢∨≠\)'"'=quoted
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
      {ω≠⍤(~∧≠\)'"'=ω}quoted
quotedtext
      {ω⊆⍤(~∧≠\)'"'=ω}quoted
```

quoted	text

Not-equal

```
      quoted ← 'extract the "quoted" parts from this "text"'
      q ← '"'=quoted    ⍝ Quotes
      p ← 2</0,≠\q      ⍝ Start of each quoted segment
      ↑ quoted q p
e x t r a c t   t h e   " q u o t e d "   p a r t s   f r o m   t h i s   " t e x t "
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
```

Not-equal

```
      (⊢⊆⍤⍥⍳∘(~∧≠\)=∘'"') quoted
```

quoted	text

```
      css ← '* {color: #000; /* text */ background: #fff; /* bg */}'
      cpos ← '/*'∘(≠\⍷⍣¯1⍳∘⊃⍷)css
      ↑ css cpos
* {color: #000; /* text */ background: #fff; /* bg */}
0000000000000000111111111100000000000000000111111110
```

Not-equal

```
      data                                    48 AAA 0.78354
48 AAA 0.78354                                70 AAA 0.55965
70 AAA 0.55965                                 5 AAA 0.96690
 5 AAA 0.96690                               54 AAA 0.17215
54 AAA 0.17215                               47 BBB 0.68116
47 BBB 0.68116                               15 BBB 0.26669
15 BBB 0.26669                               37 CCC 0.41522
37 CCC 0.41522                               91 CCC 0.32995
91 CCC 0.32995                               31 CCC 0.38465
31 CCC 0.38465
```

Not-equal

```
      4 2 3
```

```
48 AAA 0.78354
70 AAA 0.55965
 5 AAA 0.96690
54 AAA 0.17215
47 BBB 0.68116
15 BBB 0.26669
37 CCC 0.41522
91 CCC 0.32995
31 CCC 0.38465
```

Not-equal

```
      MTake 4 2 3
1 0 0 0 1 0 1 0 0
      MTake,4 2 3,⍤0⊢2
1 0 0 0 1 0 1 0 1 0 1 0 0 1 0
      ⎕←e←≠⌿MTake,4 2 3,⍤0⊢2
1 1 1 1 0 0 1 1 0 0 1 1 1 0 0
```

```
48 AAA 0.78354
70 AAA 0.55965
 5 AAA 0.96690
54 AAA 0.17215
47 BBB 0.68116
15 BBB 0.26669
37 CCC 0.41522
91 CCC 0.32995
31 CCC 0.38465
```

Not-equal

```
      ⍞←e←≠\MTake,4 2 3,¨0⊢2
1 1 1 1 0 0 1 1 0 0 1 1 1 0 0
      e⌿⍤data
48 AAA 0.78354
70 AAA 0.55965
 5 AAA 0.96690
54 AAA 0.17215


47 BBB 0.68116
15 BBB 0.26669


37 CCC 0.41522
91 CCC 0.32995
31 CCC 0.38465
```

```
48 AAA 0.78354
70 AAA 0.55965
 5 AAA 0.96690
54 AAA 0.17215
47 BBB 0.68116
15 BBB 0.26669
37 CCC 0.41522
91 CCC 0.32995
31 CCC 0.38465
```

Flat pa

STSC

Boole *dition*

```
∇ Z←P PANDRED V
[1]   Z←(V≤P)/P ◇ Z←(Z/1⌽Z)∧P/V
∇

∇ Z←P PORRED V
[1]   Z←(V∨P)/P ◇ Z←(Z/1⌽Z)≤P/V
∇

∇ Z←P PEQRED V
[1]   Z←'=' N⍀(1⌽P)/=⍀V
∇

∇ Z←P PNERED V
[1]   Z←'≠' N⍀(1⌽P)/≠⍀V
∇

∇ Z←P PLTRED V
[1]   Z←(P≥V=1⌽P)/P
[2]   Z←(Z/1⌽Z)∧P/V=1⌽P
∇

∇ Z←P PMAXRED V
[1]   Z←V[(⍒V)[P/⍋(+⍀P)[⍒V]]]
∇

∇ Z←P PMINRED V
[1]   Z←V[(⍋V)[P/⍋(+⍀P)[⍋V]]]
∇

∇ Z←P PGRADEUP V
[1]   Z←⌷⍳0+(⍋V)[⍋(+⍀P)[⍋V]]-⌈⍀P×⍳⍴P
∇

∇ Z←P PPLRED V
[1]   Z←'-' N⍀(1⌽P)/+⍀V
∇

∇ Z←P PPLREDB V
[1]   Z←((V∨P)/P),1
[2]   Z←(1+'-' N⍀ Z/⍳⍴Z)-¯P/V
∇
```

```
∇ Z←P PANDSCAN V
[1]   Z←¬≠⍀(V≤P)⍀'≠' N⍀~(V≤P)/V
∇

∇ Z←P PORSCAN V
[1]   Z←≠⍀(V∨P)⍀'≠' N⍀(V∨P)/V
∇

∇ Z←P PEQSCAN V
[1]   Z←=⍀V≠P⍀'≠' N⍀~P/≠⍀¯1↓1,V
∇

∇ Z←P PNESCAN V
[1]   Z←≠⍀V≠P⍀'≠' N⍀ P/≠⍀¯1↓0,V
∇

∇ Z←P PLTSCAN V
[1]   Z←(V∧P)∨(V∨P)⍀'>' N⍀(V∨P)/V
∇

∇ Z←P PMAXSCAN V
[1]   Z←⍋(⍋V)[⍋(+⍀P)[⍋V]]
[2]   Z←V[Z⍳⌈⍀Z]
∇

∇ Z←P PMINSCAN V
[1]   Z←⍋(⍒V)[⍋(+⍀P)[⍒V]]
[2]   Z←V[Z⍳⌈⍀Z]
∇

∇ Z←P PGRADEDOWN V
[1]   Z←⌷⍳0+(⍒V)[⍋(+⍀P)[⍒V]]-⌈⍀P×⍳⍴P
∇

∇ Z←P PPLSCAN V
[1]   Z←+⍀V-P⍀'-' N⍀ P/+⍀¯1↓0,V
∇

∇ Z←P PREVERSE V
[1]   Z←V[⊖⍒+⍀P]
∇
```

Flat partition

```
      1 0 0 0 1 0 0 {∈⍵¨⍺⊂⍵} 'ABCDXYZ'
DCBAZYX
```

Flat partition

```
      b ← 1 0 0 0 1 0 0
      a ← 'ABCDXYZ'
      ↑ a b (+\b)
A B C D X Y Z
1 0 0 0 1 0 0
1 1 1 1 2 2 2
      ⍧+\b
5 6 7 1 2 3 4
      ⌽⍧+\b
4 3 2 1 7 6 5
      a[⌽⍧+\b]
DCBAZYX
```

Flat partition

```
      p ← 1,1↓1=?1e4⍴10
      t ← ⎕A[?1e4⍴26]

      ]runtime -c "p {∊⌽¨α⊂ω} t" "p {ω[⌽⍒+\α]} t"

  p {∊⌽¨α⊂ω}    t → 1.3E¯4 |    0% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  p {ω[⌽⍒+\α]} t → 3.6E¯5 | -73% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
```

Flat partition

```
      1 0 0 0 1 0 0 {∈+/¨⍺⊂⍵} 1 2 3 4 5 6 7
10 18
```

Flat partition

```
      _P ← {∈αα¨α⊂ω}

      si ← ?1e4⍴100
      li ← ?1e4⍴1000
      f ← ?1e4⍴0

      ⎕DR¨si li f
83 163 645
```

Flat partition

```
      ]runtime -c "parts +/_P data" "parts {¯2-/0,(1⌽α)/+\ω} data"
  p +/_P si                 → 6.2E¯5 |    0% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
  p {¯2-/0,(1⌽α)/+\ω} si → 7.8E¯6 | -88% □□□□□

  p +/_P li                 → 7.0E¯5 |    0% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
  p {¯2-/0,(1⌽α)/+\ω} li → 7.8E¯6 | -89% □□□□

  p +/_P f                 → 7.1E¯5 |    0% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
* p {¯2-/0,(1⌽α)/+\ω} f → 1.6E¯5 | -77% □□□□□□□□□
```

Efficient implementation

Bit Booleans in APL **Larry Breed** circa 1966

SIMD without special SIMD hardware (SWAR)

Efficient implementation

apl.wiki/boolean

Robert Bernecky

A Compendium of SIMD Boolean Array Algorithms in APL

Sixteen Boolean reductions

f	f/⍵ is true if ⍵ satisfies:
0⍨	never
∧	all 1s
>	odd number of leading 1s
⊣	first bit is 1
<	last bit is the only 1
⊢	last bit is 1
≠	odd number of 1s
∨	at least one 1
⍱	odd number of leading 0s else last is the only 1
=	even number 0s
~⍤⊢	last is parity of the length
≥	even leading 0s
~⍤⊣	first is 0
≤	last is not the only 0
⍲	even leading 0s else last is only 0
1⍨	always

Further reading

apl.wiki/boolean

S.B. Jaffe *Topics for a Second Course in APL*

dl.acm.org/doi/10.1145/22008.22025

R. Bernecky *A Compendium of SIMD Boolean Array Algorithms*

Phil Last *Boolean Reductions*

archive.vector.org.uk/art10501140

STSC *Boolean Functions and Techniques*

Next Webinar

Thursday 15th April

16:00 BST (UTC+1)

Dyalog.TV

Error Handling - Part 2

Adám Brudzewsky

APL Seeds '21

dyalog.com/apl-seeds-user-meetings/aplseeds21.htm

Wednesday 31st March
14:00 BST (UTC+1)

Will You Play APL With Me?
How an APL Prototype Helped Designing a Service
Algorithms as a Tool of Thought
Simulation and Gaming with APL
Zoom meetup afterwards

British APL Association

britishaplassociation.org/webinar-schedule-2021

Thursday 25th March 2021

Open Session