DYALOG
Glasgow 2016

Array Oriented Functional Programming With Dyalog APL

Morten Kromberg
Roger Hui

(based on work by Jay Foad & John Scholes)

History of APL

Kenneth E. Iverson
1920-2004

- Canadian of Norwegian Descent
- Born on a small farm in Alberta (Canada)
- Finished one-room school after 9th grade and worked on the farm
- Army 1942, Flight Engineer in Air Force from 1943
  - Almost finished High School in the service
  - Promised his officers and mates that he would pursue an academic career after the war
- B.A. from Queens University, Kingston Ontario
  - Ken didn't know there was such a thing as University before he joined the army!

History of APL, continued

- Doctoral work at Harvard with Aiken and Leontief
- Taught at Harvard for 6 years,
  - frustrated with inadequacies of mathematical notation
- Developed ”Iverson Notation” in response
  - Published ”A Programming Language” in 1962

ACM Turing award in 1979:

“For his pioneering effort in programming languages and mathematical notation resulting in what the computing field now knows as APL,
for his contributions to the implementation of interactive systems, to educational uses of APL,
and to programming language theory and practice.”

Syntaxes of Mathematics

$a\,b$

$Mat1 \cdot Mat2$

Problems:

- *Wide variety of syntactical forms*
- *Strange and inconsistent precedence rules*
- *Things get worse when you deal with matrices*

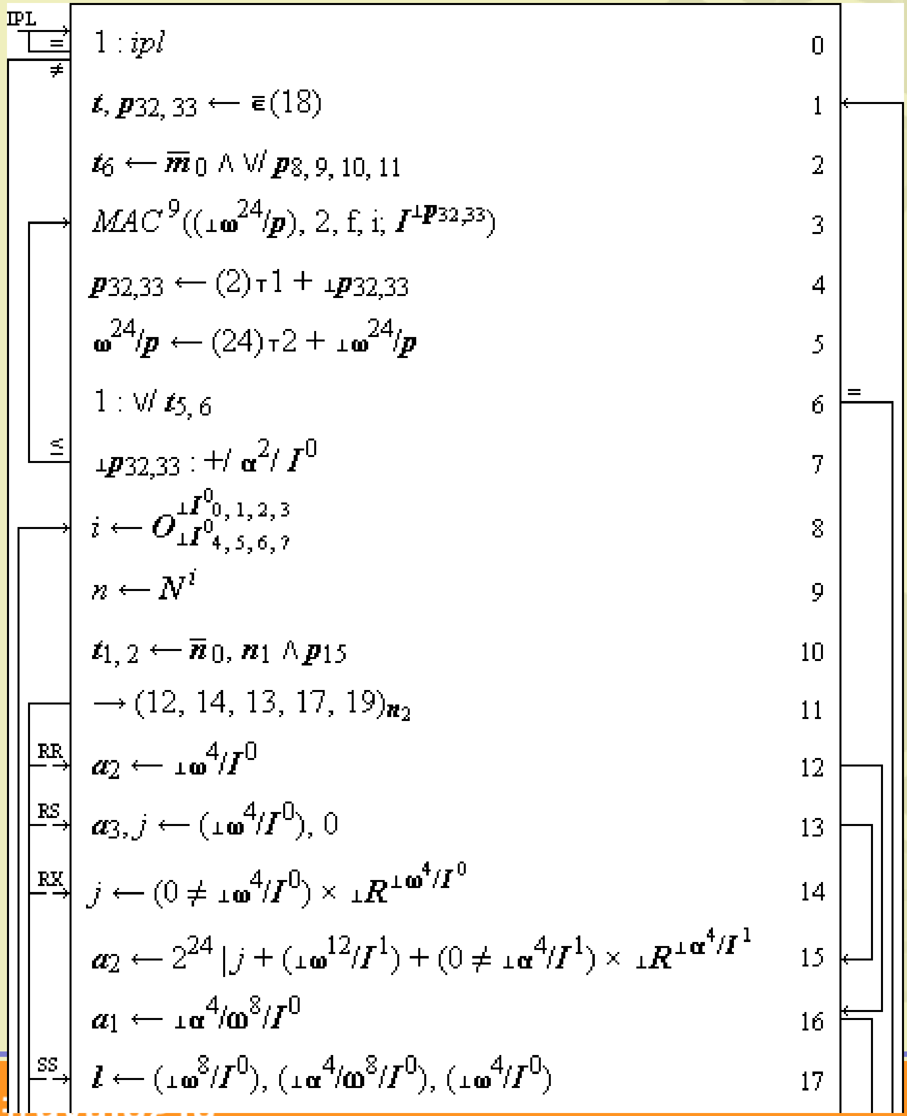
See http://www.jsoftware.com/papers/EvalOrder.htm

$\sum_{n=1} 4n$

$\prod_{i=1} 4i$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Iverson Notation: Description of IBM\360


IPL
=
≠
≤
=
RR
RS
RX
SS

$1 : ipl$ 0

$\boldsymbol{t}, \boldsymbol{p}_{32,33} \leftarrow \boldsymbol{\epsilon}(18)$ 1

$\boldsymbol{t}_6 \leftarrow \overline{\boldsymbol{m}}_0 \wedge \vee/\boldsymbol{p}_{8,9,10,11}$ 2

$MAC^9((\perp\boldsymbol{\omega}^{24}/\boldsymbol{p}), 2, \mathrm{f}, \mathrm{i}; \boldsymbol{I}^{\perp\boldsymbol{p}_{32,33}})$ 3

$\boldsymbol{p}_{32,33} \leftarrow (2)\top 1 + \perp\boldsymbol{p}_{32,33}$ 4

$\boldsymbol{\omega}^{24}/\boldsymbol{p} \leftarrow (24)\top 2 + \perp\boldsymbol{\omega}^{24}/\boldsymbol{p}$ 5

$1 : \vee/\boldsymbol{t}_{5,6}$ 6

$\perp\boldsymbol{p}_{32,33} : +/\boldsymbol{\alpha}^2/\boldsymbol{I}^0$ 7

$i \leftarrow O^{\perp\boldsymbol{I}^0_{0,1,2,3}}_{\perp\boldsymbol{I}^0_{4,5,6,7}}$ 8

$n \leftarrow N^i$ 9

$\boldsymbol{t}_{1,2} \leftarrow \overline{\boldsymbol{n}}_0, \boldsymbol{n}_1 \wedge \boldsymbol{p}_{15}$ 10

$\rightarrow (12, 14, 13, 17, 19)_{\boldsymbol{n}_2}$ 11

$\boldsymbol{a}_2 \leftarrow \perp\boldsymbol{\omega}^4/\boldsymbol{I}^0$ 12

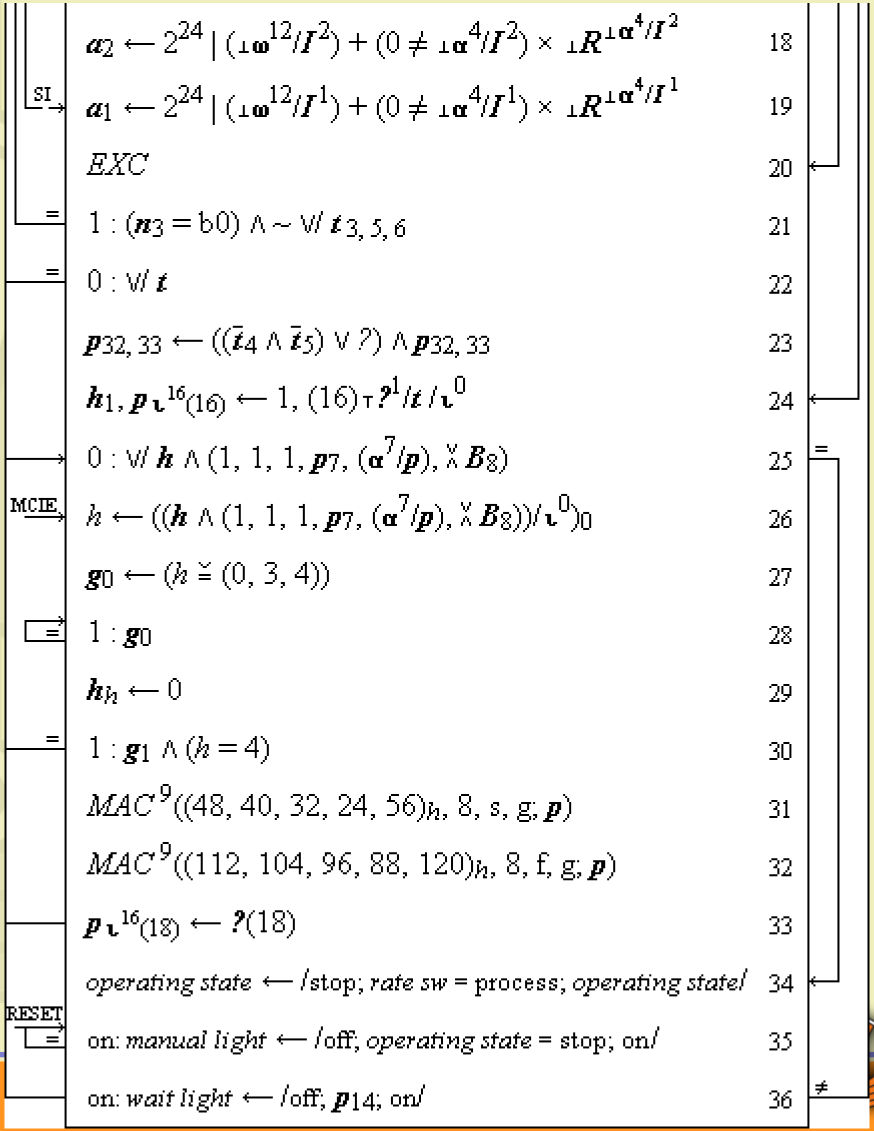
$\boldsymbol{a}_3, j \leftarrow (\perp\boldsymbol{\omega}^4/\boldsymbol{I}^0), 0$ 13

$j \leftarrow (0 \neq \perp\boldsymbol{\omega}^4/\boldsymbol{I}^0) \times \perp\boldsymbol{R}^{\perp\boldsymbol{\omega}^4/\boldsymbol{I}^0}$ 14

$\boldsymbol{a}_2 \leftarrow 2^{24} \,|\, j + (\perp\boldsymbol{\omega}^{12}/\boldsymbol{I}^1) + (0 \neq \perp\boldsymbol{\alpha}^4/\boldsymbol{I}^1) \times \perp\boldsymbol{R}^{\perp\boldsymbol{\alpha}^4/\boldsymbol{I}^1}$ 15

$\boldsymbol{a}_1 \leftarrow \perp\boldsymbol{\alpha}^4/\boldsymbol{\omega}^8/\boldsymbol{I}^0$ 16

$\boldsymbol{l} \leftarrow (\perp\boldsymbol{\omega}^8/\boldsymbol{I}^0), (\perp\boldsymbol{\alpha}^4/\boldsymbol{\omega}^8/\boldsymbol{I}^0), (\perp\boldsymbol{\omega}^4/\boldsymbol{I}^0)$ 17


SI
=
=
MCIE
=
=
RESET
=
=
≠

$\boldsymbol{a}_2 \leftarrow 2^{24} \,|\, (\perp\boldsymbol{\omega}^{12}/\boldsymbol{I}^2) + (0 \neq \perp\boldsymbol{\alpha}^4/\boldsymbol{I}^2) \times \perp\boldsymbol{R}^{\perp\boldsymbol{\alpha}^4/\boldsymbol{I}^2}$ 18

$\boldsymbol{a}_1 \leftarrow 2^{24} \,|\, (\perp\boldsymbol{\omega}^{12}/\boldsymbol{I}^1) + (0 \neq \perp\boldsymbol{\alpha}^4/\boldsymbol{I}^1) \times \perp\boldsymbol{R}^{\perp\boldsymbol{\alpha}^4/\boldsymbol{I}^1}$ 19

EXC 20

$1 : (\boldsymbol{n}_3 = \mathrm{b}0) \wedge \sim \vee/\boldsymbol{t}_{3,5,6}$ 21

$0 : \vee/\boldsymbol{t}$ 22

$\boldsymbol{p}_{32,33} \leftarrow ((\overline{\boldsymbol{t}}_4 \wedge \overline{\boldsymbol{t}}_5) \vee ?) \wedge \boldsymbol{p}_{32,33}$ 23

$\boldsymbol{h}_1, \boldsymbol{p}\boldsymbol{\iota}^{16}(16) \leftarrow 1, (16)\top \boldsymbol{?}^1/\boldsymbol{t}/\boldsymbol{\iota}^0$ 24

$0 : \vee/\boldsymbol{h} \wedge (1, 1, 1, \boldsymbol{p}_7, (\boldsymbol{\alpha}^7/\boldsymbol{p}), \overset{\vee}{\wedge}\boldsymbol{B}_8)$ 25

$h \leftarrow ((\boldsymbol{h} \wedge (1, 1, 1, \boldsymbol{p}_7, (\boldsymbol{\alpha}^7/\boldsymbol{p}), \overset{\vee}{\wedge}\boldsymbol{B}_8))/\boldsymbol{\iota}^0)_0$ 26

$\boldsymbol{g}_0 \leftarrow (h \overset{\vee}{=} (0, 3, 4))$ 27

$1 : \boldsymbol{g}_0$ 28

$\boldsymbol{h}_h \leftarrow 0$ 29

$1 : \boldsymbol{g}_1 \wedge (h = 4)$ 30

$MAC^9((48, 40, 32, 24, 56)_h, 8, \mathrm{s}, \mathrm{g}; \boldsymbol{p})$ 31

$MAC^9((112, 104, 96, 88, 120)_h, 8, \mathrm{f}, \mathrm{g}; \boldsymbol{p})$ 32

$\boldsymbol{p}\boldsymbol{\iota}^{16}(18) \leftarrow \boldsymbol{?}(18)$ 33

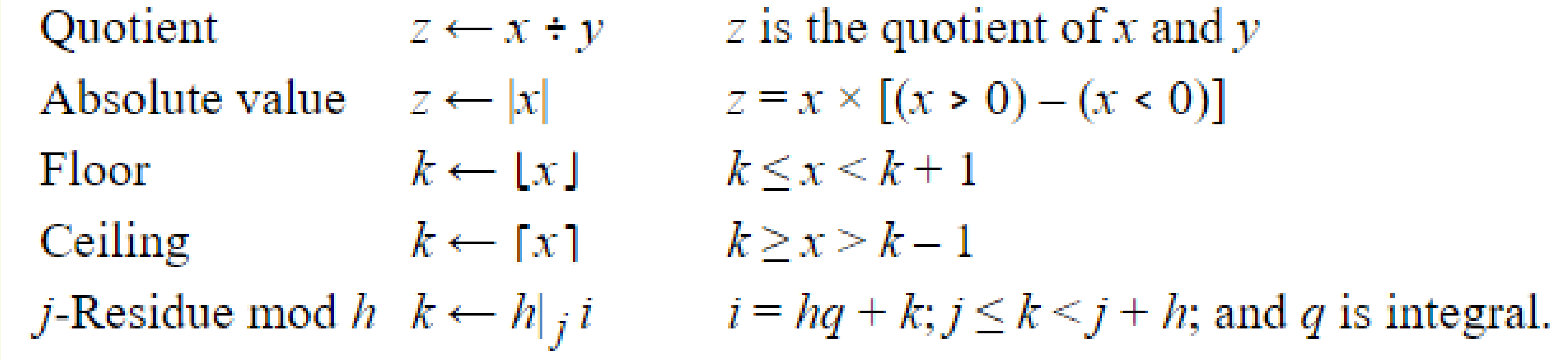
operating state ← /stop; *rate sw* = process; *operating state*/ 34

on: *manual light* ← /off; *operating state* = stop; on/ 35

on: *wait light* ← /off; $\boldsymbol{p}_{14}$; on/ 36

A Programming Language

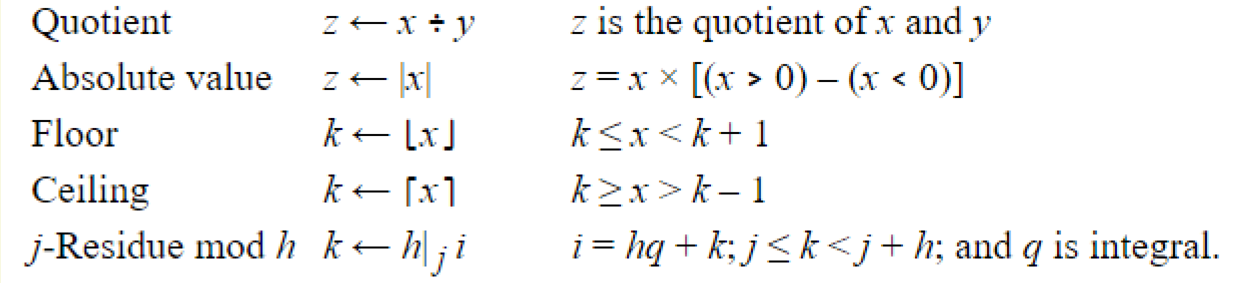
- The book, 1962



Quotient	$z \leftarrow x \div y$	z is the quotient of x and y
Absolute value	$z \leftarrow \vert x \vert$	$z = x \times [(x > 0) - (x < 0)]$
Floor	$k \leftarrow \lfloor x \rfloor$	$k \leq x < k + 1$
Ceiling	$k \leftarrow \lceil x \rceil$	$k \geq x > k - 1$
j-Residue mod h	$k \leftarrow h \vert_j i$	$i = hq + k; j \leq k < j + h$; and q is integral.

Linearization => APL\360

- The 5: Ken Iverson, Adin Falkoff, Larry Breed, Dick Lathwell, Roger Moore. Operated by “Quaker Consensus”.



Quotient	$z \leftarrow x \div y$	z is the quotient of x and y
Absolute value	$z \leftarrow \lvert x \rvert$	$z = x \times [(x > 0) - (x < 0)]$
Floor	$k \leftarrow \lfloor x \rfloor$	$k \leq x < k + 1$
Ceiling	$k \leftarrow \lceil x \rceil$	$k \geq x > k - 1$
j-Residue mod h	$k \leftarrow h\rvert_j\, i$	$i = hq + k; j \leq k < j + h$; and q is integral.

The first saved workspace:

```
    )LOAD 1 CLEANSPACE
SAVED  1966-11-27  15.53.59 (GMT-7)
```

A Programming Language (for Mathematics)

a×b

ab

Mat1 +.× Mat2

$Mat1 \cdot Mat2$

*x

e^x

f g x

$f\,g\,x$

x÷y

$\frac{x}{y}$

(f+g) x

$(f+g)\,x$

(3○x)*2

$\tan^2 x$

b⍟a

$\log_b a$

a*÷n

$\sqrt[n]{a}$

+/4×⍳6

$\sum_{n=1}^{6} 4n$

×/4×⍳6

$\prod_{i=1}^{6} 4i$

$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

(2×a)÷⍨(-b)(+,-)0.5*⍨(b*2)-4×a×c

Saving Your Work

- Historically, APL users have saved workspaces containing code and data as a single file
  - Similar to an Excel Spreadsheet
  - Takes a ”snapshot of the VM”
  - Beware: also saves the execution stack if there is one
- Save your work using

```
      )save /path/mywsname[.dws]
```

- Load it again with

```
      )load /path/mywsname
```

- You can extract named objects from a workspace:

```
      )copy /path/mywsname foo goo x y
```

- Saved workspaces can have a ”latent expression” ⎕LX, which is executed when the workspace is loaded, unless you

```
      )xload /path/mywsname
```

Saving Your Work, continued

- In the last few years, it has become more popular to use Unicode text files (and SVN/GIT), especially for source code
- You can save a fn/var, namespace or class using `]save` (`]save` is a ”user command” - written in APL):

  ```
      ]save name /path/name[.dyalog]
  ```

  (it is customary to use the same name for the file)
- You can bring it back into the active workspace using

  ```
      ]load /path/name
      ]load /path/*
  ```

- If you edit objects that were `]load`ed, the system will offer to update the file each time you make a change
- From version 15.0, the interpreter (editor) knows how to open and view source files without user commands.

Reading APL

```
CB←{⍵[1+(⍴⍵)|X∘.+X←(⍳⍺)-1]}
```

Imagine arguments: `8 CB ' ⎕'`

Reading APL

```
life←{↑1 ω∨.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂ω}
```

http://dfns.dyalog.com/n_life.htm
http://www.youtube.com/watch?v=a9xAKttWgP4

- Try it out in small chunks, starting from the right.
- Know what's a function and what's an operator.
- Note the creative use of inner product.
- Finally try reading it from left to right.

Quirks that you may notice:

- Some APL programmers like to avoid parentheses, to reduce the cognitive load!
- Hence, put simple argument on left: 1+...
- Or, use Commute: 2*⍨...
- N.B. game of life has no parentheses, partly because (some) primitives (e.g. Residue) were carefully designed to be most useful with a simple constant on the *left*.

Reading APL

What does this function do?

```
{(~R∊R∘.×R)/R←1↓⍳⍵}
```

Or:

```
{{(~⍵∊⍵∘.×⍵)/⍵}1↓⍳⍵}
```

“Procedures”

- Before dfns, APL had an imperative form now known as “tradfns”
- The only control flow was → (goto)
- Control structures (`:If` etc) arrived in the late 1980’s
- Within “tradfns”, names have dynamic scope

Procedures / Tradfns

Monadic:

```
      ∇ R←Sum X
[1]     R←+/X
      ∇
```

Dyadic:

```
      ∇ R←A MatMult B
[1]     R←A+.×B
      ∇
```

Niladic:

```
      ∇ Run
[1]     ⎕←'Boo Hiss!'
      ∇
```

Procedures / Tradfns

”Ambi-valent” (+ use a control structure)

```
      ∇ R←{Window} Sum X
[1]     :If 0=⎕NC 'Window' ⋄ R←+/X
[2]     :Else ⋄ R←Window +/ X
[3]     :EndIf
      ∇
```

Procedures / Tradfns

Name Elements of Right Argument

+ Local Variable

+ Class / DotNet declarations

```
    ∇ r←Round(n decimals);base
      :Access Public Shared
      :Signature Double←Round Double N, Int32 Decimals
      base←10*decimals
      r←(⌊0.5+n×base)÷base
    ∇
```

Errors

```
      1 2 3÷4 5
LENGTH ERROR
      1 2 3÷4 5
     ^
      ⎕EN
5
      1÷0
DOMAIN ERROR: Divide by zero
      1÷0
     ^
      ⎕EN (⎕EM 11)
 11  DOMAIN ERROR
      ⎕DMX.Message
Divide by zero
```

Errors

1	WS FULL
2	SYNTAX ERROR
3	INDEX ERROR
4	RANK ERROR
5	LENGTH ERROR
6	VALUE ERROR
7	FORMAT ERROR
10	LIMIT ERROR
11	DOMAIN ERROR
12	HOLD ERROR
13	OPTION ERROR
15	LST FULL
16	NONCE ERROR
17	ACCESS ERROR
18	FILE TIE ERROR
19	FILE ACCESS ERROR
20	FILE INDEX ERROR
21	FILE FULL
22	FILE NAME ERROR
23	FILE DAMAGED
24	FILE TIED
25	FILE TIED REMOTELY
26	FILE SYSTEM ERROR
28	FILE SYSTEM NOT AVAILABLE
30	FILE SYSTEM TIES USED UP
31	FILE TIE QUOTA USED UP
32	FILE NAME QUOTA USED UP
34	FILE SYSTEM NO SPACE
35	FILE ACCESS ERROR - CONVERTING FILE
36	INCOMPATIBLE ARRAY
38	FILE COMPONENT DAMAGED

Error Trapping: Dfns

```
div←{0::'Something Else is Wrong'
    11::0 ⍝ DOMAIN error: return 0
    ⍺÷⍵}
      3 div 0
0
      1 2 3 div 4 5
Something Else is Wrong
```

Error Trapping: Tradfns

Using :Trap

```
     ∇ R←A DIV B
[1]    :Trap 0
[2]        R←A÷B
[3]    :Case 11 ⋄ R←0 ⍝ DOMAIN error
[4]    :Else ⋄ R←'Something Else is Wrong'
[5]    :EndTrap
     ∇
```

Error Trapping: Tradfns

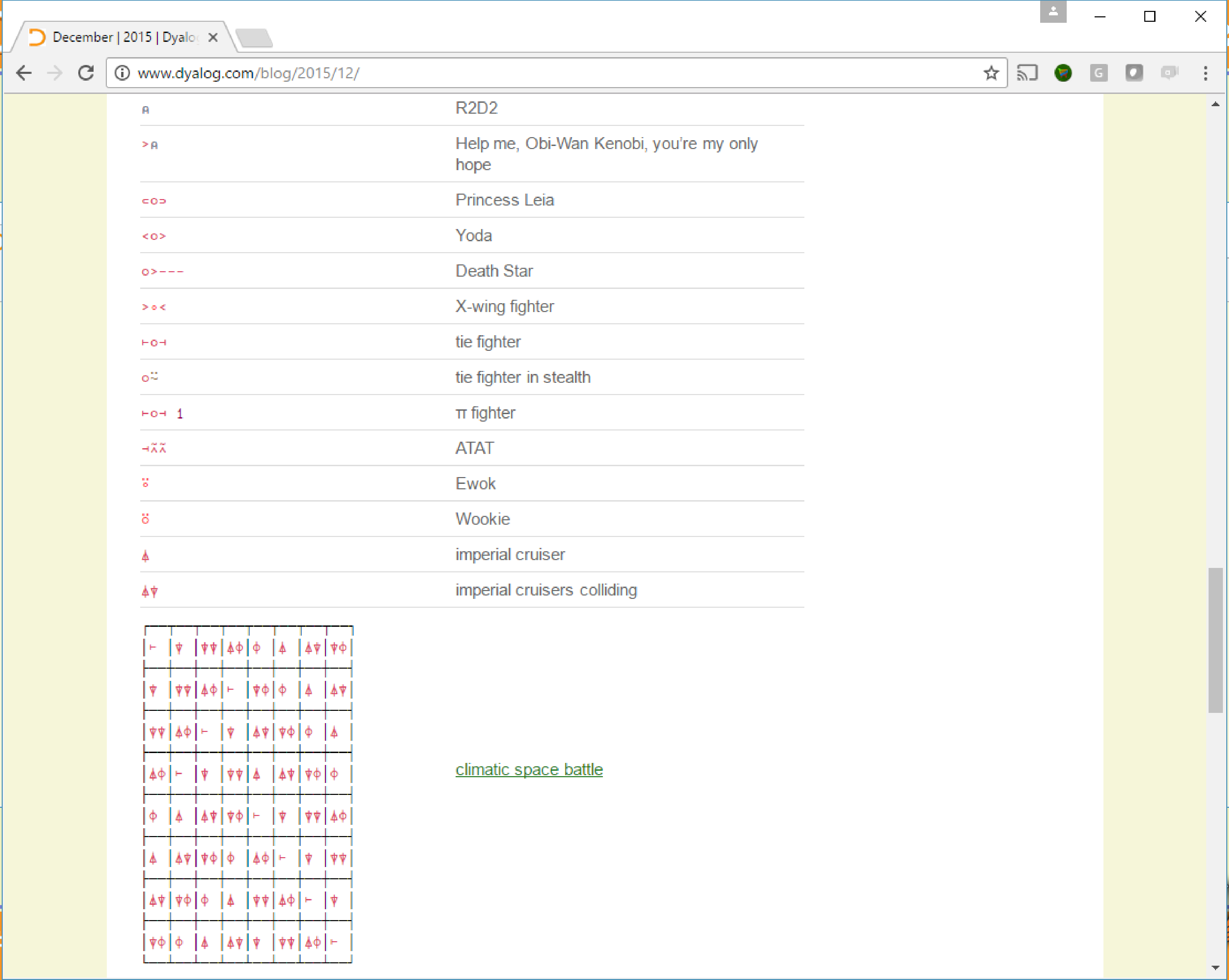
Using ⎕TRAP

```
∇ R←A DIVQ B;⎕TRAP

  ⎕TRAP←(11 'E' '→DOMERR')(0 'C' '→CATCHALL')
  R←A÷B ⋄ →0

 DOMERR:→R←0
 CATCHALL:R←'Something Else is Wrong'
∇
```

Building Applications

Delivery Mechanisms:

- Saved Workspace
- Workspace Bound with Interpreter as an .exe
- COM Server
- Microsoft.Net Assembly
  - ASP.Net scripting language, Web Services, SharePoint Web Parts, WCF, etc etc etc etc
- “Stand Alone” Web Server, Web Service



⍝	R2D2
>⍝	Help me, Obi-Wan Kenobi, you're my only hope
⊂○⊃	Princess Leia
<○>	Yoda
○>---	Death Star
>∘<	X-wing fighter
⊢○⊣	tie fighter
○⍨	tie fighter in stealth
⊢○⊣ 1	π fighter
⊣⍲⍲	ATAT
⍤	Ewok
⍥	Wookie
⍋	imperial cruiser
⍋⍒	imperial cruisers colliding

```
┌──┬──┬──┬──┬──┬──┬──┬──┐
│⊢ │⍒ │⍒⍒│⍋⌽│⌽ │⍋ │⍋⍒│⍒⌽│
├──┼──┼──┼──┼──┼──┼──┼──┤
│⍒ │⍒⍒│⍋⌽│⊢ │⍒⌽│⌽ │⍋ │⍋⍒│
├──┼──┼──┼──┼──┼──┼──┼──┤
│⍒⍒│⍋⌽│⊢ │⍒ │⍋⍒│⍒⌽│⌽ │⍋ │
├──┼──┼──┼──┼──┼──┼──┼──┤
│⍋⌽│⊢ │⍒ │⍒⍒│⍋ │⍋⍒│⍒⌽│⌽ │
├──┼──┼──┼──┼──┼──┼──┼──┤
│⌽ │⍋ │⍋⍒│⍒⌽│⊢ │⍒ │⍒⍒│⍋⌽│
├──┼──┼──┼──┼──┼──┼──┼──┤
│⍋ │⍋⍒│⍒⌽│⌽ │⍋⌽│⊢ │⍒ │⍒⍒│
├──┼──┼──┼──┼──┼──┼──┼──┤
│⍋⍒│⍒⌽│⌽ │⍋ │⍒⍒│⍋⌽│⊢ │⍒ │
├──┼──┼──┼──┼──┼──┼──┼──┤
│⍒⌽│⌽ │⍋ │⍋⍒│⍒ │⍒⍒│⍋⌽│⊢ │
└──┴──┴──┴──┴──┴──┴──┴──┘
```

climatic space battle