DYALOG

Elsinore 2023

Tacit Techniques

Adám Brudzewsky

Rich Park

asst. Peter Mikkelsen

DYALOG

Elsinore 2023

These Tacit Techniques Will Blow Your Mind!

APL Wiki	apl.wiki/tacit#Tutorials
Adám Brudzewsky	xpqz.github.io/cultivations/Trains
Rich Park	youtu.be/Enlh5qwwDuY “Train Spotting”

What is the essence of tacit programming? Find out here!

Explicit code mentions arguments:

- Expression `(⌈/N)-⌊/N←3 1 4 1 5`
- Tradfn `∇ R←Range Y …`
- Dfn `{(⌈/⍵)-(⌊/⍵)}`

Tacit code implies arguments:

- Tacit `⌈/-⌊/`

Shocking Revelation: You've been unknowingly using tacit constructs all along!

```
f/    f⍨    ∘.g    f\    A∘g    f⍤    f⍤B
```

APL old-timers don't want you to know this one fact:

Operator application is actually just tacit programming!

Function composition

```
f⍤g    f⍥g    f⍨    fgh    f∘g
```

Discover the mind-blowing secret:

Function composition is actually just plumbing!

Experts rave about tacit!

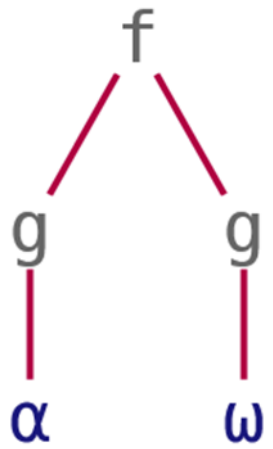
- Arguments in operands
- Memorable (like ≠⊆⊢ and +⌿÷≠)
- Adjacency (like ×- and ∨/⍷)
- Brevity (like F⍤⍥C)
- DRY (Don’t Repeat Yourself; like ≡⍤⍴)
- Just a general feeling of superiority and awesomeness

Over

The shape of an outer product `α ∘.f ω` is
`(⍴α) , (⍴ω)`

We can write this as
`α ,⍤⍴ ω`

“pre-process both”



`α f⍤g ω`

Beside

Location of α^{th} 1 in each element of ω is

```
α ⊃¨ ⍸¨ ω
```

We can write this as

```
α ⊃∘⍸¨ ω
```

“pre-process right”


```
α f∘g ω
```

Atop

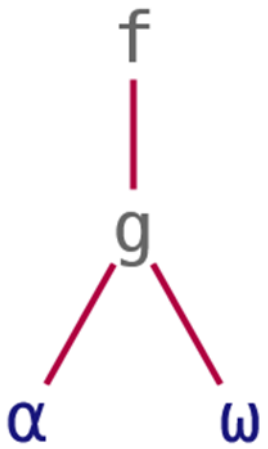
Any-presence of α in ω is

```
∨/ α ∈ ω
```

We can write this as

```
α ∨/⍤∈ ω
```

“post-processing result”




```
α f⍤g ω
```

Commute

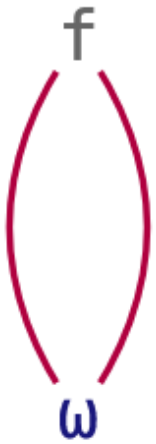
A multiplication table of N is

```
(⍳⍵) ∘.× (⍳⍵)
```

We can write this as

```
∘.×⍨ ⍳⍵
```

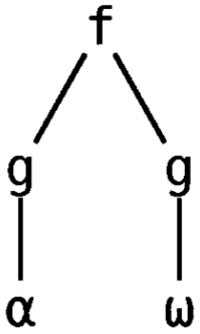
“selfie”




```
f⍨ ⍵
```

Tacit Techniques
Function Compositions

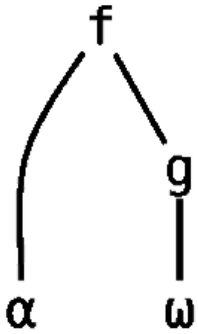
Over



α f⍥g ω

pre-process both

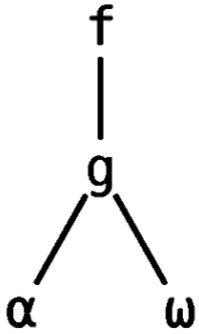
Beside



α f∘g ω

pre-process right

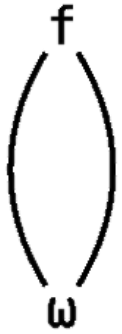
Atop



α f⍤g ω

post-process result

Commute



f⍨ ω

selfie

Tasks: Tacify!

1. `'Hello' {(⎕C α)≡(⎕C ω)} 'HELLO'` `1`
2. `'ab' 'cd' {∧/α∈ω}¨ 'abba' 'dad'` `1 0`
3. `64 1000 0 {α*÷ω} 3` `4 10 0`
4. `10 4 1 0 {α××ω} ¯3 2 0 ¯1` `¯10 4 0 0`

Train Introduction

Also a type of composition

Sequence of functions in isolation:

- Parenthesised:

```
(+⌿÷≢) 3 1 4 1 5
```

- Assigned

```
Avg←+⌿÷≢
Avg 3 1 4 1 5
```

Discussion: Writing Trains

```
(f Y) + (h Y)
```

Discussion: Writing Trains

```
(f Y) + (h Y)   →   (f + h) Y
(f Y) + (  Y)
```

Discussion: Writing Trains

```
(f Y) + (h Y)   →   (f + h) Y
(f Y) + (  Y)   →   (f + ⊢) Y
```

```
(X f Y) + (X h Y)
```

Discussion: Writing Trains

```
(f Y) + (h Y)   →   (f + h) Y
(f Y) + (  Y)   →   (f + ⊢) Y
```

```
(X f Y) + (X h Y)   →   X (f + h) Y
(X    ) + (X h Y)
```

Discussion: Writing Trains

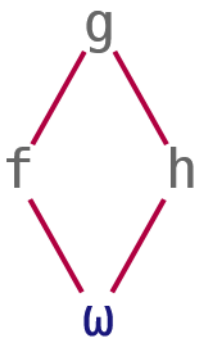
```
(f Y) + (h Y)   →   (f + h) Y
(f Y) + (  Y)   →   (f + ⊢) Y
```

```
(X f Y) + (X h Y)   →   X (f + h) Y
(X    ) + (X h Y)   →   X (⊣ + h) Y
```

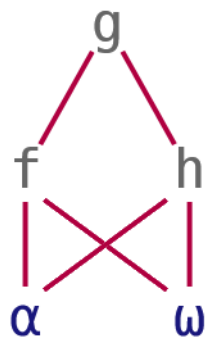
Discussion: Writing Trains

```
(f Y) g (h Y)   →   (f g h) Y
(f Y) g (  Y)   →   (f g ⊢) Y
       g (h Y)
```

```
(X f Y) g (X h Y)   →   X (f g h) Y
(X     ) g (X h Y)   →   X (⊣ g h) Y
```



(fgh)ω



α(fgh)ω

Discussion: Writing Trains

```
(f Y) g (h Y)   →   (f g h) Y
(f Y) g (  Y)   →   (f g ⊢) Y
      g (h Y)   →   (  g h) Y

(X f Y) g (X h Y)   →   X (f g h) Y
(X    ) g (X h Y)   →   X (⊣ g h) Y
      g (X h Y)
```

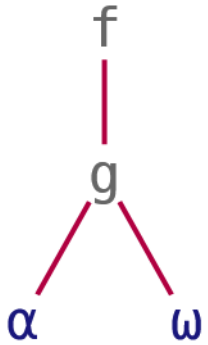
Discussion: Writing Trains

```
(f Y) g (h Y)   →   (f g h) Y
(f Y) g (  Y)   →   (f g ⊢) Y
      g (h Y)   →   (  g h) Y

(X f Y) g (X h Y)   →   X (f g h) Y
(X   ) g (X h Y)   →   X (⊣ g h) Y
       g (X h Y)   →   X (  g h) Y
```

Discussion: Writing Trains

```
(f Y) g (h Y)   →   (f g h) Y
(f Y) g (  Y)   →   (f g ⊢) Y
      f (g Y)   →   (  f g) Y

(X f Y) g (X h Y)   →   X (f g h) Y
(X    ) g (X h Y)   →   X (⊣ g h) Y
      f (X g Y)   →   X (  f g) Y
```

(fg)ω



α(fg)ω

Train Details

Some parts can actually be arrays:

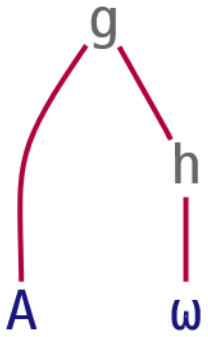
- `A g h` is `{A} g h`

Two sub-types of trains:

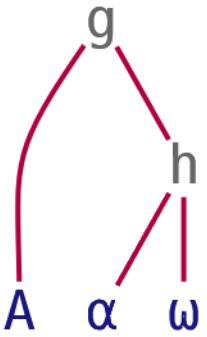
- Fork: `f g h` and `A g h`
- Atop: `f g`

Longer trains:

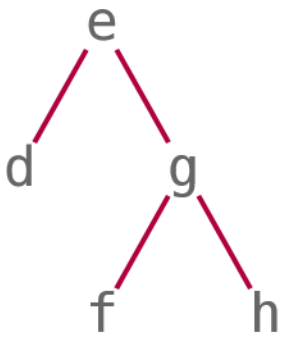
- `(d e f g h)` is `(d e (f g h))`
- `(e f g h)` is `(e (f g h))`

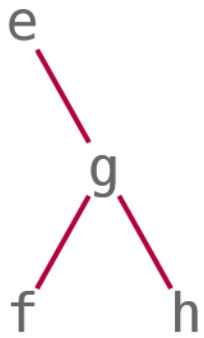


`(Agh)ω`



`α(Agh)ω`

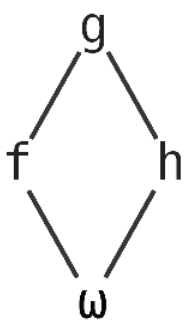




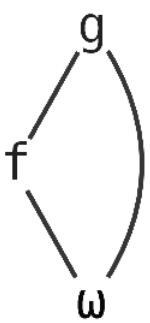
Tacit Techniques
Trains

Monadic

```
(f Y) g (h Y)  ↔  (f g h) Y
(f Y) g (  Y)  ↔  (f g ⊢) Y
       g (h Y)  ↔  (  g h) Y
```



(fgh)ω

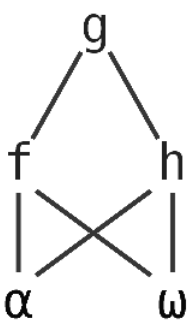


(fg⊢)ω

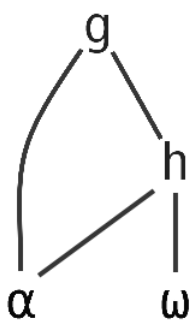
(fg)ω

Dyadic

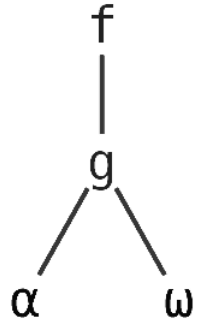
```
(X f Y) g (X h Y)  ↔  X (f g h) Y
(X    ) g (X h Y)  ↔  X (⊣ g h) Y
       g (X h Y)  ↔  X (  g h) Y
```



α(fgh)ω



α(⊣gh)ω



α(fg)ω

Example: Writing a Train

```
      {∧/ (⌈\⍵) = ⍵}  1 3 5 6 7
1
      {∧/ (⌈\⍵) = (⊢⍵)}  1 3 5 6 7
1
      {∧/ (⌈\ = ⊢) ⍵}  1 3 5 6 7
1
      {(∧/ ⌈\ = ⊢) ⍵}  1 3 5 6 7
1
      (∧/ ⌈\ = ⊢)  1 3 5 6 7
1
```

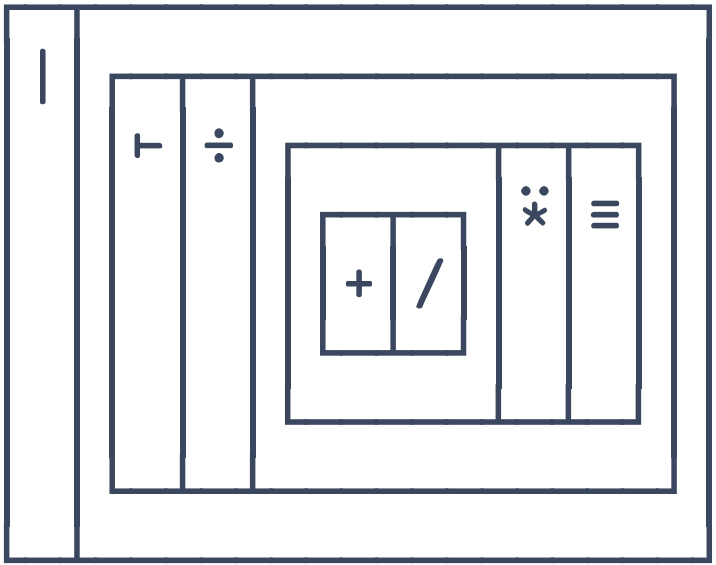
Tasks: Tacify!

1. `{2×ω} 2 7 1 8` → `4 14 2 16`
2. `3 1 4 {(α∪ω)~(α∩ω)} 1 6 1` → `3 4 6`
3. `{∪ω∨⍳ω} 10` → `1 2 5 10`
4. `2 {(α⊃∀ω)⊃ω} 3 1 4 1 5` → `4`

Amazing tool: `]box on`

```
]box on
```

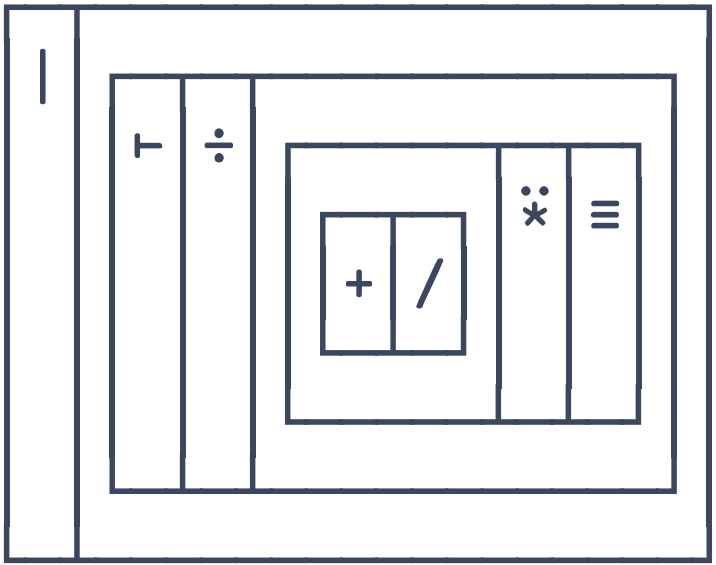
```
|⊢÷+/⁖≡
```



```
]box on -t=…
```

```
|⊢÷+/⁖≡
```

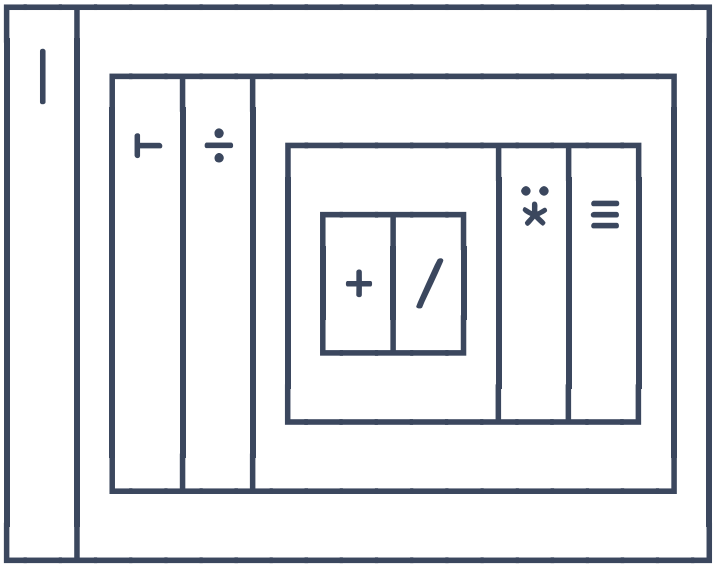
```
-t=box
```



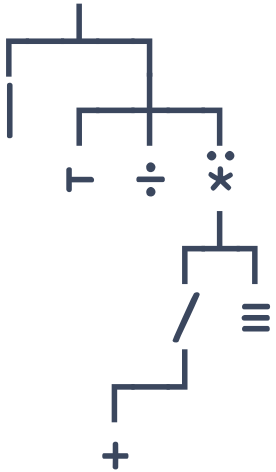
]box on -t=…

|⊢÷+/⁖≡

-t=box



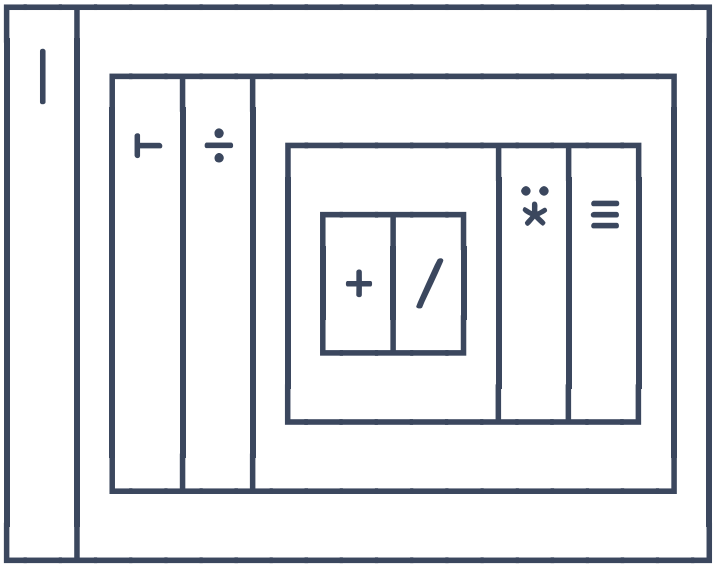
-t=tree



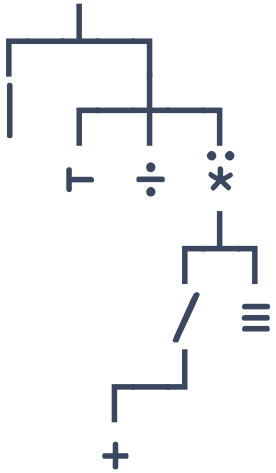
]box on -t=…

|⊢÷+/⁖≡

-t=box



-t=tree



-t=parens

```
|(⊢÷((+/)⁖≡))
```

Tasks: Tacify!

1. `{(⊖⍵)⊥(⊖⍵)} 1 1 1 0 1 1 0 0` → `3`
2. `',;' {(~⍵∈⍺)⊆⍵} 'ab,de;fgh'` → `ab de fgh`

```
nums←3 1 4 1 5
```

3. `4 {(⍺+≠⍵)÷⍺} nums` → `2.25 2.75`
4. `{(+≠⍵)÷≢⍵} nums` → `2.8`
5. **Bonus task:** Combine 3 & 4 into an ambivalent function.

4 aspects of tacit that could ruin your life – #4 will blow your mind!

- Arguments in operands
- Monadic functions
- Recursion, Assignment, Dotting
- Selection (but 20.0…)

The ultimate paradox revealed: Arguments in operands

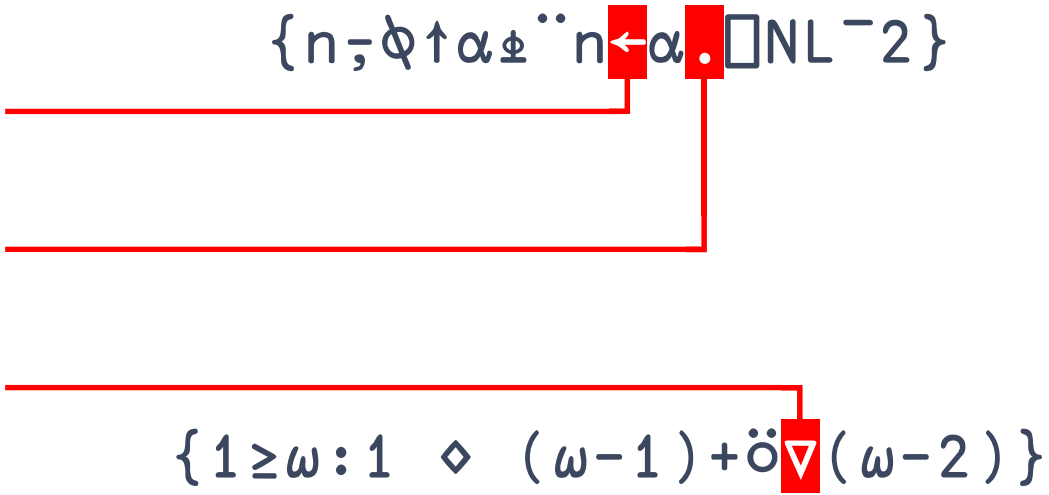
```
      10 {x←α ⋄ ϕ@{x<ω}ω} 1 2 13 14 5 16 7 18
1 2 18 16 5 14 7 13
      10 {      ϕ@(α<⊢)ω} 1 2 13 14 5 16 7 18
1 2 18 16 5 14 7 13

      2 {(ϕ⍤⍉⍤*α)ω} 3 3⍴⍳9
9 8 7
6 5 4
3 2 1
```

Inappropriate tacit discovered: Lots of monadic functions

```
{⍵+≠↑⍵¨⍵}
⍵⍤0(+≠)⍤0↑⍤0(⍵¨)
⍵(+≠(↑⍵¨))
⍵+≠⍤0↑⍤0(⍵¨)
⍵(+≠⍤0↑⍵¨)
```

3 things tacit code just cannot do

- Assignment
- Namespace “dotting”
- Recursion



The ugly truth about selection, and what we plan to do about it

```
      {(3>⍵)≠⍵} 3 1 4 1 5
4 5
       (3∘>≠⊢)  3 1 4 1 5
SYNTAX ERROR
       (3∘>≠ ⊢) 3 1 4 1 5
SYNTAX ERROR
     (3∘> ⊢⍤≠ ⊢)3 1 4 1 5
4 5
        3∘>⍥≠   3 1 4 1 5
```

The ugly truth about selection, and what we plan to do about it

```
      'aeiou' {ω[⍺⍋ω]} 'hello world'
eoohll wrld
      'aeiou' {ω[]⍨⍺⍋ω} 'hello world'
LENGTH ERROR
      'aeiou' {ω[]⍨⊂⍺⍋ω} 'hello world'
eoohll wrld
      'aeiou'  (⊂⍨⍋[]⊢)  'hello world'
eoohll wrld
      'aeiou'   (⍋⍥⊢)   'hello world'
```

Tasks: Convert tacit to dfn

1. Monadic `××⌊⍤⍥|`
2. Dyadic `⌊∘≢↑⊢`
3. Monadic `≡⍤(⍞C~∘' ')`
4. Monadic `+⌿⊢>+⌿÷≢`

Bonus tasks:

5. Monadic `⍋≡⊢⊢≡⍋`

Amazing tool: tacit.help

Transform tacit APL into dfn form

```
f ← +⌿÷⊣∘≢
```

```
f Y ⇔ {((+⌿)⍵)÷(≢⍵)}
```

```
X f Y ⇔ {(⍺(+⌿)⍵)÷⍺}
```

Arrays: A, B, C,… Functions: a, b, c,…

7 hash tables that won’t go away

```
P ⍋ s
P ⍒ s
P ⍳ s
P ∪ s
s ∈ P
s ~ P
s ∩ P
```

```
P∘⍋ s
P∘⍒ s
P∘⍳ s
P∘∪ s
(∈∘P) s
(~∘P) s
(∩∘P) s
```

7 hash tables that won’t go away

```
      s←'Hello, World!'
      AVi←⍞AV⍤⍳ ◇ {⍵}AVi s
      cmpx '⍞AV⍳s' 'AVi s'
⍞AV⍳s → 6.0E¯7 |   0% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
AVi s → 3.7E¯7 | -39% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
      AVg←⍞AV⍤⍋ ◇ {⍵}AVg s
      cmpx '⍞AV⍋s' 'AVg s'
⍞AV⍋s → 2.3E¯6 |   0% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
AVg s → 5.5E¯7 | -77% ⎕⎕⎕⎕⎕
```

7 hash tables that won’t go away

```
      s←'Hello, World!'

      cmpx '⎕AV⍳s' '⎕AV∘⍳ s'
⎕AV⍳s    → 6.5E¯7 |   0% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
⎕AV∘⍳ s → 7.0E¯7 | +7% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕

      cmpx '⎕AV⍋s' '⎕AV∘⍋ s'
⎕AV⍋s    → 2.0E¯6 |     0% ⎕
⎕AV∘⍋ s → 6.7E¯5 | +3242% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
```

Showdown: Memory vs CPU

```
      ⍪e
```

`a (∧/⍤∈⍤) b`
`a {∧/⍤α∈⍤ω} b`
`a ∧/⍥∈⍤ b`
`a {∧/α∈ω}⍤ b`

```
a←(2e4⍴⍤'ab' 'cd')[?5e4⍴2]
b←     'aba' 'cad'[?5e4⍴2]

∇ hwm←mem expr
  {}⎕WA
  {}0(2000⌶)14
  hwm←2000⌶14
  {}⍎expr
  hwm-⍨←2000⌶14
∇
```

Showdown: Memory vs CPU

```
      (⍪⍤,,mem¨) e
a (∧/¨∈¨) b     125835728
a {∧/¨α∈¨ω} b   125835792
a ∧/⍤¨∈¨ b        4194304
a {∧/α∈ω}¨ b      4194304
      (⍪⍤,,1∘cmpx¨) e
a (∧/¨∈¨) b     0.45675
a {∧/¨α∈¨ω} b   0.4145
a ∧/⍤¨∈¨ b      0.4245
a {∧/α∈ω}¨ b    0.42275
```

Showdown: Memory vs CPU

```
      (⍪⍤¯1,mem¨) e
a (∧/¨∊¨) b          0
a {∧/¨α∊¨ω} b        0
a ∧/⍤∊¨ b      4194304
a {∧/α∊ω}¨ b   4194304
      (⍪⍤¯1,1∘cmpx¨) e
a (∧/¨∊¨) b    0.0080625
a {∧/¨α∊¨ω} b  0.0078359375
a ∧/⍤∊¨ b      0.019703125
a {∧/α∊ω}¨ b   0.021765625
```

```
a←2↑¨a
```

Check out this cool summary — you won't regret it!

Function composition:

- ⍥ *Pre-process both*
- ∘ *Pre-process right*
- ⍤ *Post-process*
- ≍ *Selfie*

Operators: long left scope

Trains: odd-even from right

Tools: `]box on -t=…`
tacit.help

Watch out for these:

Arguments in operands

Lots of monadic functions

Just don't try:

- Assignment
- Namespace "dotting"
- Recursion

Selection issues:

- Compress: ⊢⍥≠
- Index: ⊂⍥… ⌷ …