

Hi





APL

APL

Ideas

- Precedence rules
- Array oriented
- Concise
- Functions
- Interactive environment

$$10 \div 2 \times 4 + 1 = ?$$

$$10 \div 2 \times 4 + 1 \neq 21$$

$$10 \div (2 \times (4 + 1)) = 1$$

$$10 \div 2 \times 4 + 1 = 1$$

Array Oriented



$$1 + 4 = 5$$

$$2 + 5 = 7$$

$$3 + 6 = 9$$

$$1\ 2\ 3 + 4\ 5\ 6 = 5\ 7\ 9$$

Concise

- Symbols!
- Composition

Symbols!

+

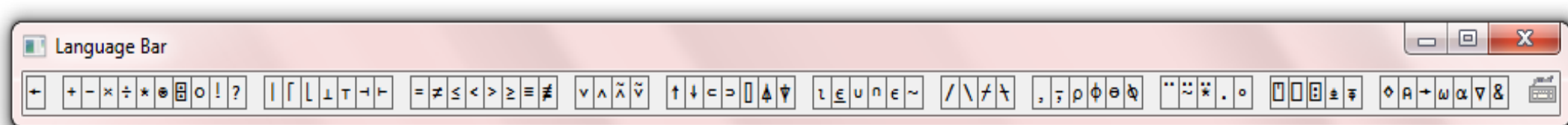
-

×

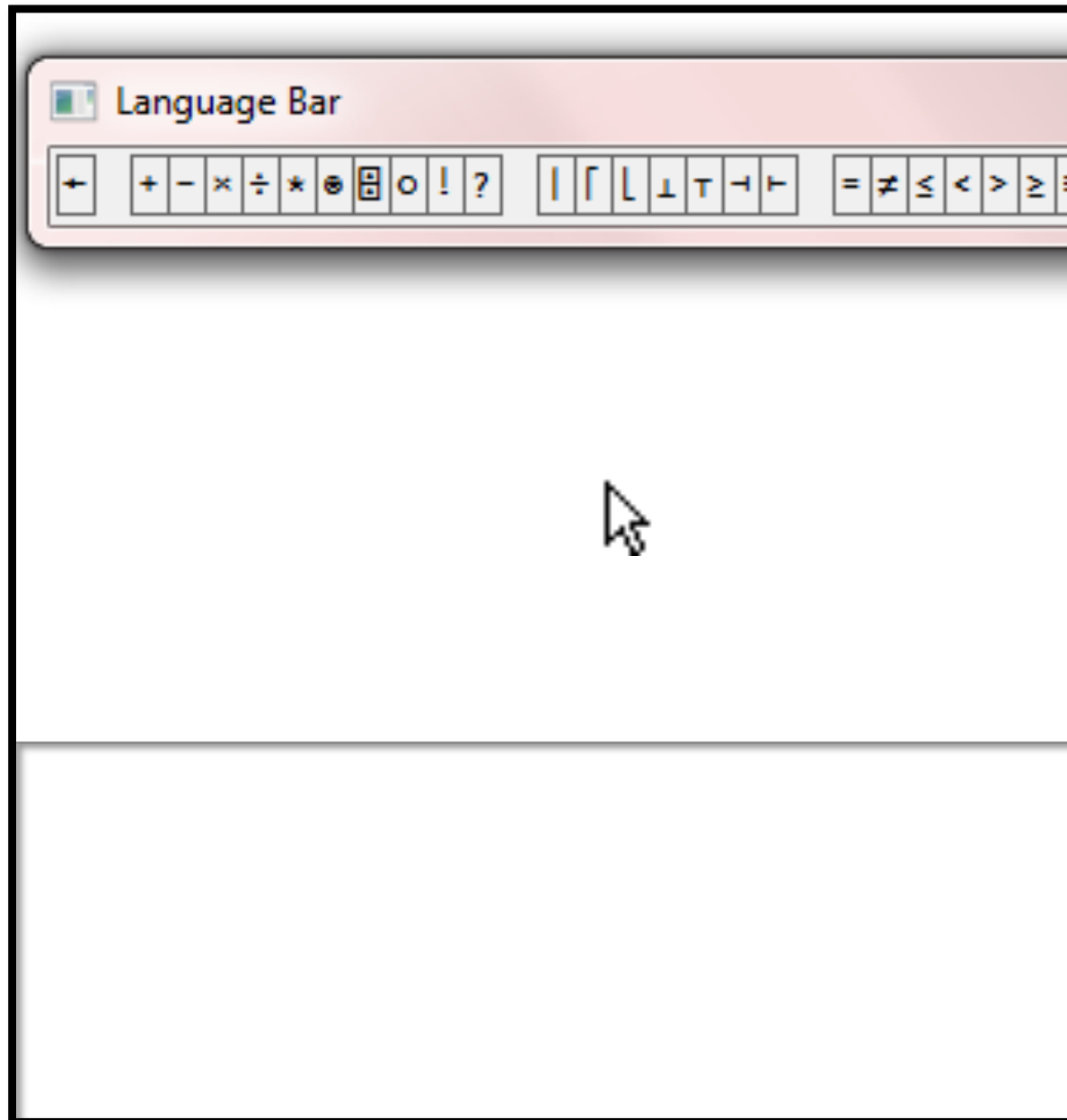
÷

Bonus Feature!

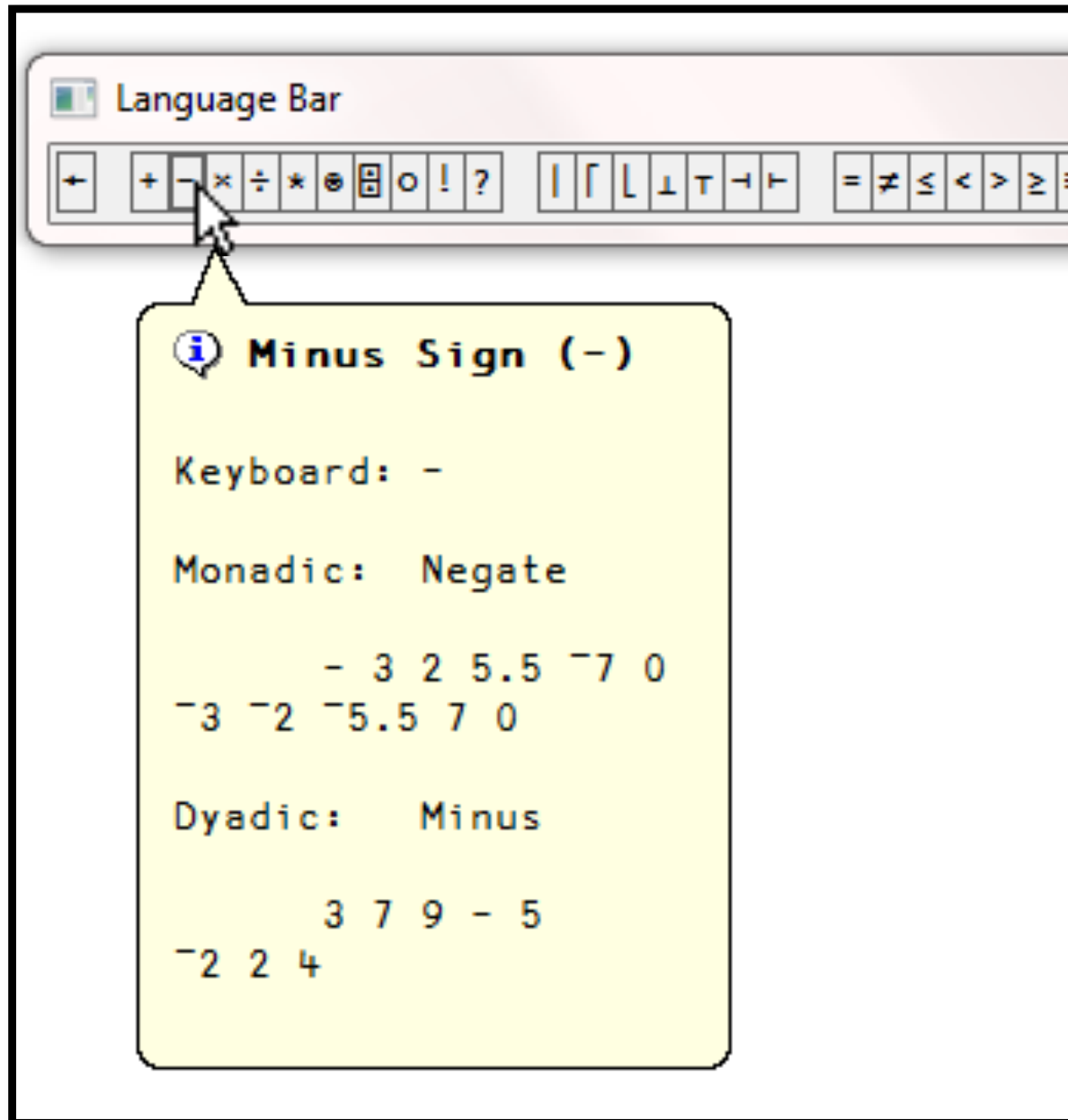
-Language Bar-



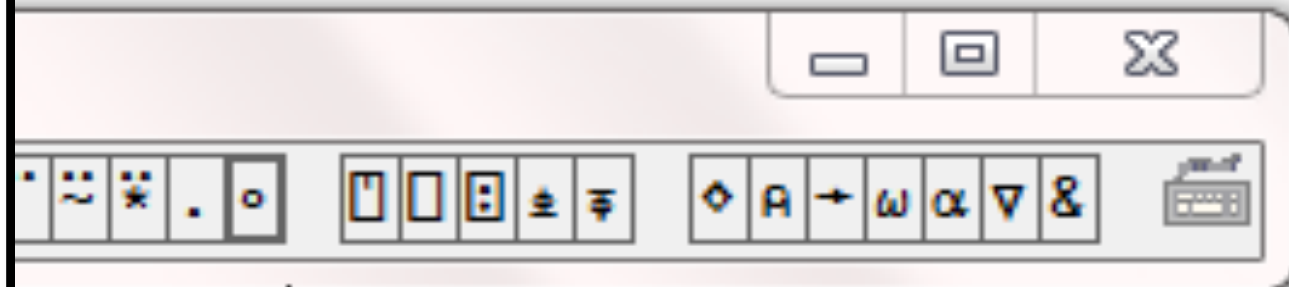
In case of “Symbol Crisis”



In case of “Symbol Crisis”



Composition



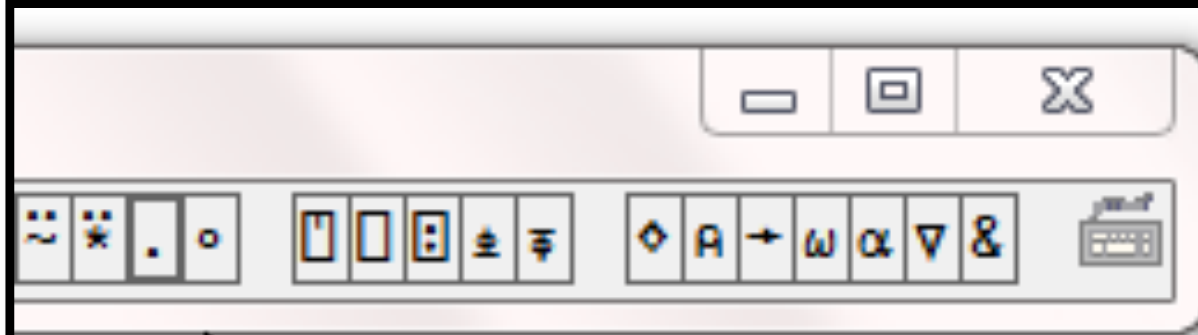
i Jot (◦)

Keyboard: Control+J

Operator: Composed With / Outer Product

```
      (*◦0.5) 1 2 3 4 5 6
1 1.414 1.732 2 2.236 2.449
```

```
      1 2 3◦.×4 5 6
4 5 6
8 10 12
12 15 18
```



i Inner Product (.)

Keyboard: .

Operator: Inner Product / Outer Product

1 2 3+.*4 5 6

32

1 2 3o.*4 5 6

4 5 6

8 10 12

12 15 18

1 2 3 \circ \cdot \times 4 5 6

1 2 3 ◦ . × 4 5 6

<u>4</u>		

1 2 3 ◦ × 4 5 6

4	<u>5</u>	

1 2 3 ◦ .x 4 5 6

4	5	<u>6</u>

1 2 3 ◦ . x 4 5 6

4	5	6
<u>8</u>		

1 2 3 ◦ . x 4 5 6

4	5	6
8	<u>10</u>	

1 2 3 \circ . x 4 5 6

4	5	6
8	10	<u>12</u>

1 2 3 ◦ . × 4 5 6

4	5	6
8	10	12
<u>12</u>		

1 2 3 ◦ . × 4 5 6

4	5	6
8	10	12
12	<u>15</u>	

1 2 3 ◦ . × 4 5 6

4	5	6
8	10	12
12	15	<u>18</u>

1 2 3 \circ \cdot \times 4 5 6

4	5	6
8	10	12
12	15	18

1 2 3 \circ \cdot \times 4 5 6

4	5	6
8	10	12
12	15	18

1 2 3 \circ . + 4 5 6

1 2 3 ◦ + 4 5 6

<u>5</u>		

1 2 3 ◦ + 4 5 6

5	<u>6</u>	

1 2 3 ◦ + 4 5 6

5	6	<u>7</u>

1 2 3 ◦ + 4 5 6

5	6	7
<u>6</u>		

1 2 3 ◦ + 4 5 6

5	6	7
6	<u>7</u>	

1 2 3 ◦ + 4 5 6

5	6	7
6	7	<u>8</u>

1 2 3 ° . + 4 5 6

5	6	7
6	7	8
<u>7</u>		

1 2 3 ° . + 4 5 6

5	6	7
6	7	8
7	<u>8</u>	

1 2 3 ° + 4 5 6

5	6	7
6	7	8
7	8	<u>9</u>

1 2 3 \circ . + 4 5 6

5	6	7
6	7	8
7	8	9

1 2 3 ° ., 4 5 6

1 2 3 ◦ ., 4 5 6

<u>1,4</u>		

1 2 3 ° 1 4 5 6

1,4	<u>1,5</u>	

1 2 3 ◦ ., 4 5 6

1,4	1,5	<u>1,6</u>

1 2 3 ◦ ., 4 5 6

1,4	1,5	1,6
<u>2,4</u>		

1 2 3 ◦ 4 5 6

1,4	1,5	1,6
2,4	<u>2,5</u>	

1 2 3 ◦ 4 5 6

1,4	1,5	1,6
2,4	2,5	<u>2,6</u>

1 2 3 ° 4 5 6

1,4	1,5	1,6
2,4	2,5	2,6
<u>3,4</u>		

1 2 3 ° 4 5 6

1,4	1,5	1,6
2,4	2,5	2,6
3,4	<u>3,5</u>	

1 2 3 ° ., 4 5 6

1,4	1,5	1,6
2,4	2,5	2,6
3,4	3,5	<u>3,6</u>

1 2 3 ◦ ., 4 5 6

1,4	1,5	1,6
2,4	2,5	2,6
3,4	3,5	3,6

1 2 3 + . × 4 5 6

1 2 3 + . x 4 5 6

4

1 2 3 + x 4 5 6

4

10

1 2 3 + . x 4 5 6

4

10

18

1 2 3 +.× 4 5 6

4

10

+

18

32

1 2 3 + . × 4 5 6

32

Anonymous functions

$$2 = 10 \div 5$$

$$2 = 10 \{ \alpha \div \omega \} 5$$

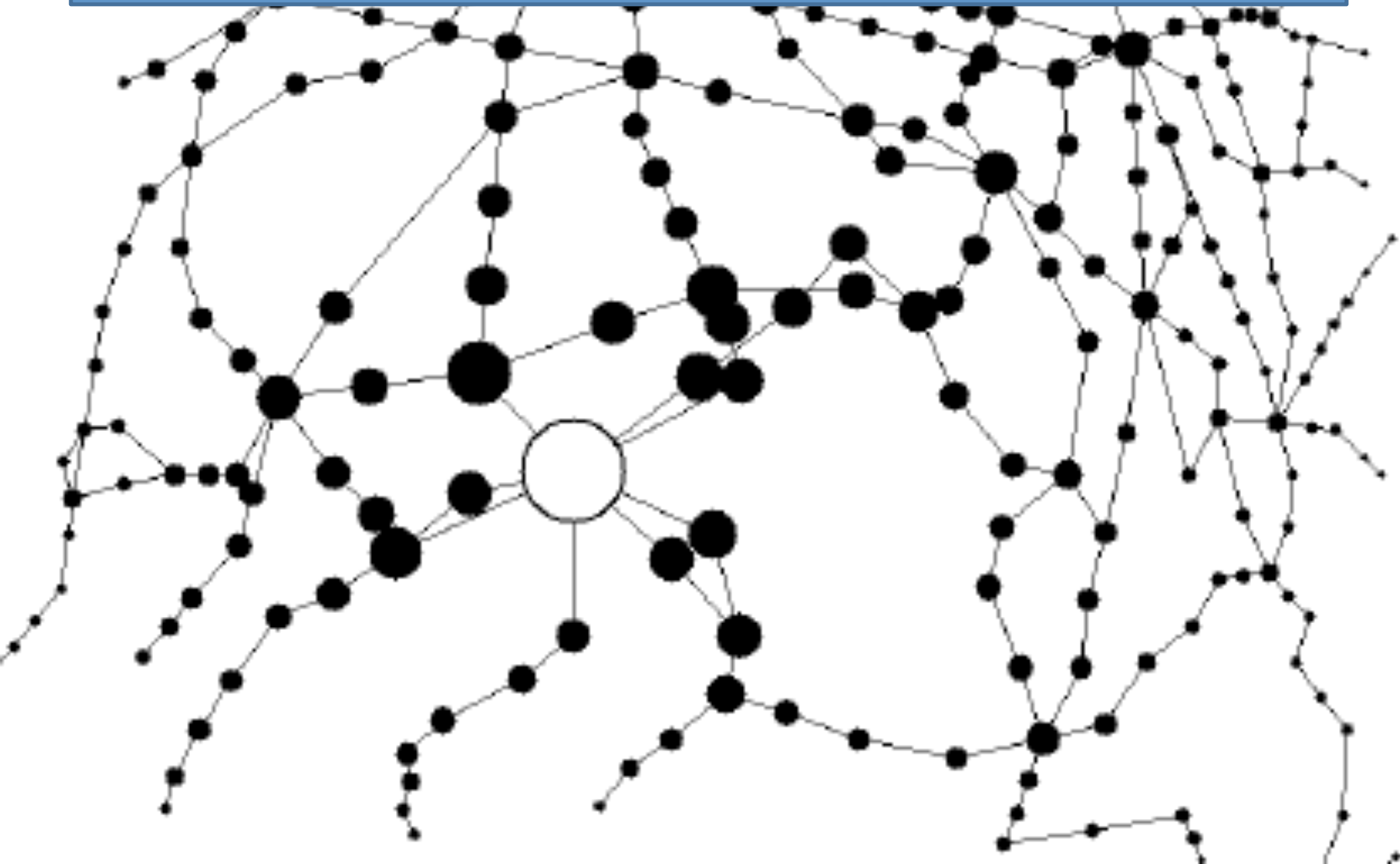
$$14 = 10 \{ \alpha + (8 \times (\omega \div \alpha)) \} 5$$

Assignment

DividedBy $\leftarrow \{a \div \omega\}$



I wonder...



Flights

To

	LAX	SFO	SAN	OAK
LAX	0	1	0	0
SFO	1	0	1	0
SAN	0	1	0	0
OAK	0	0	1	0

From

OneHop←Flights

	To				
	LAX	SFO	SAN	OAK	
From	LAX	0	1	0	0
	SFO	1	0	1	0
	SAN	0	1	0	0
	OAK	0	0	1	0

TwoHops ← OneHop + . × OneHop

	To			
	LAX	SFO	SAN	OAK
From LAX	1	0	1	0
From SFO	0	2	0	0
From SAN	1	0	1	0
From OAK	0	1	0	0

ThreeHops ← OneHop + . × TwoHops

To

	LAX	SFO	SAN	OAK
LAX	0	2	0	0
SFO	2	0	2	0
SAN	0	2	0	0
OAK	1	0	1	0

AllTrips ← OneHop + TwoHops + ThreeHops

To

	LAX	SFO	SAN	OAK
LAX	1	3	1	0
SFO	3	2	3	0
SAN	1	3	1	0
OAK	1	1	2	0

From

v \wedge $\tilde{\wedge}$ \tilde{v}

↑ ↓ = > □ △ ▽

ℓ ∈ ∪ ∩ ∈ ~

/ \ ≠ †

, ; ρ φ θ φ

“ ∴

i Logical And (\wedge)

Keyboard: Control+0

Dyadic: Logical And / Lowest Common Multiple

0 1 0 1 \wedge 0 0 1 1
0 0 0 1

15 1 2 7 \wedge 35 1 4 0
105 1 4 0

+ . ×

V. ^

TwoHops \leftarrow OneHop + \times OneHop
To

	LAX	SFO	SAN	OAK
LAX	1	0	1	0
SFO	0	2	0	0
SAN	1	0	1	0
OAK	0	1	0	0

TwoHops \leftarrow OneHop \vee \wedge OneHop
To

	LAX	SFO	SAN	OAK
LAX	1	0	1	0
SFO	0	1	0	0
SAN	1	0	1	0
OAK	0	1	0	0

ThreeHops ← OneHop ∨ TwoHops
To

	LAX	SFO	SAN	OAK
LAX	0	1	0	0
SFO	1	0	1	0
SAN	0	1	0	0
OAK	1	0	1	0

AllTrips ← OneHop ∨ TwoHops ∨ ThreeHops

To

	LAX	SFO	SAN	OAK
LAX	1	1	1	0
SFO	1	1	1	0
SAN	1	1	1	0
OAK	1	1	1	0

ThreeHops ← OneHop ∨ OneHop ∨ .∧ TwoHops

NextHops ← PrevHops ∨ PrevHops ∨ .∧ ThisHops

NextHops ← PrevHops {α ∨ α ∨ .∧ ω} ThisHops

ThreeHops ← OneHop ∨ OneHop ∨ TwoHops

NextHops ← PrevHops ∨ PrevHops ∨ ThisHops

NextHops ← PrevHops {α ∨ α ∨ ω} ThisHops

AllTrips ← OneHop ({α ∨ α ∨ ω} * 3) OneHop

AllTrips ← OneHop ({α ∨ α ∨ ω} * ≡) OneHop

AllTripsFor ← {ω ({α ∨ α ∨ ω} * ≡) ω}

AllTripsFor $\leftarrow \{\omega (\{\alpha \vee \alpha \vee . \wedge \omega\} * \equiv) \omega\}$

Or $\leftarrow \vee$

ConnectedTo $\leftarrow \vee . \wedge$

UntilNoChange $\leftarrow \{\alpha \alpha \alpha * \equiv \omega\}$

AppliedTo $\leftarrow \{\omega \alpha \alpha \omega\}$

AllTripsFor $\leftarrow (\{\alpha \text{ Or } \alpha \text{ ConnectedTo } \omega\} \text{ UntilNoChange}) \text{ AppliedTo}$

AllTripsFor Flights

What I covered

- Precedence rules
- Array oriented
- Concise
- Functions
- Interactive environment

Things you didn't see

- Memory allocation
- For loops
- A source file
- Types



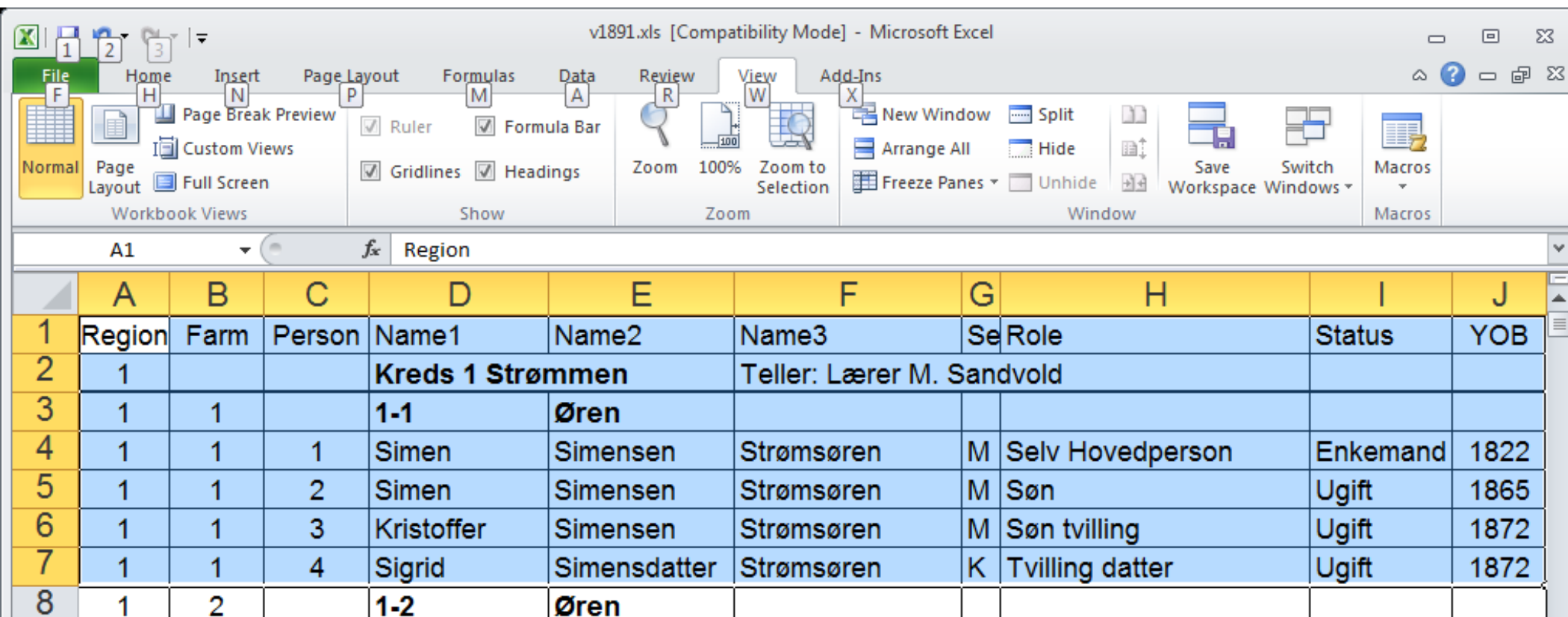
Joel@JoelHough.com

The old guy with no hat ...

- Morten Kromberg, CTO of Dyalog Ltd.
 - Dyalog is the leader of the [still] emerging APL Market 😊
 - Basingstoke (UK) + Canada, DK, France, USA
- 1 year Z80, 1 year Commodore BASIC, 33 yrs APL
 - IBM 5100 APL, SHARP APL, IBM VSAPL and APLSV, DEC APL/SF, MIPS APL (Prime), Data General APL, IBM APL2, APL*PLUS/PC, APL+Win, and now Dyalog APL
- Wrote at least one program that I understood in each of:
 - 6502 and Z80 Machine code, JCL
 - Simula, Pascal, COBOL, C, C#, Java
- ... (Plus one program in Prolog)
- CTO of Adaytum, BI "startup" based on APL
 - Sold to Cognos in 2000 for \$160M

Interactive Demo

- A Day in the Life of a Domain Expert
 - Domain: History
- Build a DSL for Playing with Tables
- Do something that might be hard in SQL



v1891.xls [Compatibility Mode] - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J
1	Region	Farm	Person	Name1	Name2	Name3	Se	Role	Status	YOB
2	1			Kreds 1 Strømmen		Teller: Lærer M. Sandvold				
3	1	1		1-1	Øren					
4	1	1	1	Simen	Simensen	Strømsøren	M	Selv Hovedperson	Enkemand	1822
5	1	1	2	Simen	Simensen	Strømsøren	M	Søn	Ugift	1865
6	1	1	3	Kristoffer	Simensen	Strømsøren	M	Søn tvilling	Ugift	1872
7	1	1	4	Sigrid	Simensdatter	Strømsøren	K	Tvilling datter	Ugift	1872
8	1	2		1-2	Øren					

What I Hope to Show You...

- That APL is an inherently parallel notation which makes it easy to think about, and “crunch”, arrays
- That APL encourages the construction of functional DSL’s – without demanding a purely functional approach to coding
- That APL is a “tool of thought”, which helps us discover suitable algorithms and solutions through experimentation

What I Hope I Demonstrated...

- The Data is at your Fingertips™
- APL Encourages the construction of functional DSLs
 - Work closely with - or BE - the Domain Expert
- ... That Dyalog APL blends Array, Functional and Object Oriented paradigms "seamlessly"
 - Allows you to use the Tool of Thought which is appropriate to the task
- ... with tight interop with Object Frameworks
 - Provide, Consume and Extend COM, CLR (& soon JVM) Classes

Parallelism

- "Map" is the default behaviour for many primitives
- We automagically parallelize each function in

+ / Quantity × Price ÷ Discount

- IBM APL2 used the "370 Vector Facility" from 1987
- However, on modern hardware, "interpreted SIMD" hits memory bottlenecks hard

Parallelism ...

- We are working on a compiler and considering "optional type declarations", etc.
- Many "idioms" are already recognized by the interpreter. For example $X/\iota \rho X$.
- We have an experimental interpreter w/closures
- P.Each P.Outer P.Rank are multi-process versions of familiar APL "operators"
 - Refactoring can be handled by non-technical developers
 - Insurance company saw 5x speedup on 8-core i7 for pension calculations
 - "bang for the buck" here and now

Can an interpreter be FAST?

- You can **always** write a faster program in a compiled language...
 - IF you have the time to figure out how to do it right
 - On large data, APL expressions have VERY LITTLE interpreter overhead – very few loops
 - $(+/\omega \div \rho \omega)$ can be thought of as "bytecode"
 - We have spent 30 years optimizing 1-bit, 1-, 2- and 4-byte integers, so arrays are compact
 - ... and memory access is getting expensive
 - Some parallelism is automatic

In Practice ...

- On synthetic benchmarks, C[#]&family usually beat APL
- ... But large APL applications often outperform *and* "outscale" competitors using compiled languages
- Close collaboration between domain experts and software engineers allows efficient operations on compact data representations
 - In rare cases hotspots are recoded in C & friends after "discovery" in APL (APL code lives on for R&D and QA)
- It is straightforward to experiment and discover alternative algorithms, and apply mathematical insights
- Object hierarchies seem to encapsulate data in ways which severely constrain the set of possible optimizations

Types and Correctness

- APL is called "write only" by those who can not read it, in practice the reverse is true for well-written code
 - Compare $\{+/\omega\div\rho\omega\}$ to traditional alternatives
- No type definitions, almost no loops or other "fluff"
- Extremely high "semantic density" leads to Domain Experts being able to read the code and participate in refactoring
- Build in hours, refactor & ship frequently - for decades 😊
- The really interesting bug is the broken algorithm

"The only program that stands a chance of being correct is a short one."

(Arthur Whitney, Kx Systems)

Some Customers

- SimCorp (DK), APL Italiana (I),
Fiserv Investment Services (US), Infostroy Ltd (Russia)
 - Leaders in various markets for Asset Management Systems
- KCI Corp (US)
 - Budgeting and Planning
- Carlisle Group (US)
 - Collateral and Securitization for Global Capital Markets
- ProfDoc Care (Sweden)
 - One of the worlds largest Patient Journal systems with 40,000 users and 2.5 million patient records in Sweden, and sub-second response
- ExxonMobil (US)
 - Optimizes the “Cracking” of Petroleum Products using APL
- Typically used in emerging or constantly churning markets

Curious?

- Download free "Mastering Dyalog APL" PDF
(or buy hardcopy from Amazon)
- Download a Free Trial System
 - Windows 32-bit,
Contact Dyalog for trials on other platforms

→ [HTTP://TRYAPL.ORG](http://tryapl.org)