Introduction to the J Programming Language

Wayne Chang

June 17, 2014

Table of contents

History
- What is J?
- Why J?
- Spiritual Successor to APL

The Language
- Terminology and Examples

What is J?

J is a programming language that is:

- Functional
  - Higher-order functions
  - Immutable Data
- Dynamic
  - Dynamically typed
  - Parsing tightly coupled with evaluation
- Array-based
  - Arrays (matrices) are first-class citizens
  - Operators can all operate in higher dimensions
- Tacit
  - Most operations are function composition
  - Higher-order function everywhere
  - Learn where parameters go

Why J?

- Operations on Matrices
- Mathematical and Statistical Programming
- Signal Processing
- Examples
  - Financial and Insurance Risk Calculations
  - Network Flow Analysis
  - Polling Institutions
  - Econometrics

Spiritual Successor to APL

- Stands for "A Programming Language"
- Developed in the 1960s by Kenneth E. Iverson
- Not a Von Neumann language (isomorph of the machine)
- Conway's Game of Life in one line of APL:
  $\Phi'\square',\in\mathbb{N}\rho\subset\mathbb{S}\leftarrow'\leftarrow\square\leftarrow(3=\mathbb{T})\vee\mathbb{M}\wedge 2=\mathbb{T}\leftarrow\supset+/(\mathbb{V}\Phi''\subset\mathbb{M}),(\mathbb{V}\ominus''\subset\mathbb{M}),(\mathbb{V},\Phi\mathbb{V})\Phi''(\mathbb{V},\mathbb{V}\leftarrow 1\ ^-1)\ominus''\subset\mathbb{M}'$
- J was developed in the early 1990's by Ken Iverson and Roger Hui

Terminology: Nouns, Verbs, Adverbs, Operators

- Right-to-left evaluation
- Nouns are data like 100, “hello, world!”, and 1.333
- Verbs are ordinary functions like $+$ or $-$
- Adverbs and operators are functions that compute functions from functions

Monads and Dyads

- Monads are verbs that take one argument on the right: $*\ 4$
- Dyads are verbs that take one argument each side: $2 + 2$
- The same symbol has multiple meanings depending on the context

Symbol	**Monadic**	**Dyadic**
$+$	Conjugate	Plus
$-$	Negate	Minus
$<.$	Integral Floor	Lesser of
$\lvert$	Magnitude	Residue

Arrays and Boxes

- Different from arrays in other programming languages.
- A table is a 2-dimensional array.
- Arrays have rank and shape.
- Cue array example: shape, tally $\#$, index of/integers (i.), join (,), from ({), match ($-$:)
- Cue box example: open ($>$), box ($<$)

Examples

- sum, product (introducing the / operator)
- vector magnitude (introducting bonding with &)
- taxes and percentages (introducing hooks)
- mean (introducing forks)
- 1D kernel convolution
- stateless FizzBuzz
- various euler problems
- introspection with 5!:4, 7!:2, 6!:2