Data Representation in S

Allan R. Wilks
AT&T Labs - Research

Introduction

- ideas percolating in my early years
- John recently back from "building 5" with QPE
- code thread independent of version 2
- main new idea: functions represented as language objects
- up to then functions were foreign code
- should be the core of version 3
- how to integrate the code threads?
- one problem: somewhat awkward data representation

Representation Issues

- binary vs. ascii
- supported structures
  - APL -- matrix
  - LISP -- atom or list
  - C -- atom or array or structure or union
- which atomic types supported
- metadata support
- impact on subsets

Version 2 Representation

- homogeneous vector of positive length
- vector elements are (non-object) atoms or other vectors
- atoms are inaccessible directly
- recursion: modal heterogeneity
  - function argument lists
  - function multiple returns
  - metadata, esp. structural
- get APL's matrices and LISP's lists
- representation is awkward, though

Internals

- internal representation:

```
name mode length *values
```

- mode is atomic or STRUCTURE for a resursive list: simple!
- but leads to this circumlocution for matrix:

```
("x" STRUCTURE 2
   ("Dim" INTEGER 2 3,4)
   ("Data" INTEGER 12 1..12)
)
```

- problems:
  - □ object-name association
  - □ Data special status
  - □ internal code has special cases

Version 3 Representation

- small change:

```
mode length *values attributes
```

- drop the name and add the attributes object
- this solves all three problems
- also added zero-length vectors and complex atomic type

Impact

- does it matter?
- both forms are capable of general representation
- but second form simplifies internal code
  - no spurious name conflicts or constraints
  - no special cases for Data component
- more importantly it simplifies user model
- led to thinking about names, dimnames and ultimately data frames

Lessons

- data representation is crucial
- flexible user community permitted change
- company release reticence may have helped!
- boundary cases are hard!
    - zero-length vectors
    - vector as row or column