J9.18

OBJECT-ORIENTED HANDLING OF NUMERICAL DATA FOR SCIENTIFIC ANALYSIS AND VISUALIZATION ¾ BASIC IDEA AND IMPLEMENTATION FOR RUBY

Takeshi Horinouchi and Naoki Kawanabe
Radio Science Center for Space and Atmosphere, Kyoto University

1. INTRODUCTION

Kinds and size of data an atmospheric scientist typically deals with have been increasing rapidly as computer power and international data exchange/sharing grow. To manage this situation, it would be desirable for him or her to have software or a programming library with which different kinds of data can be treated efficiently in a consolidated way. The consolidation may be achieved by making use of the object-oriented way to separate data and accessors to them. We propose a framework to realize it and implement it with the object-oriented language Ruby.

2. FRAMEWORK FOR GRIDDED PHYSICAL DATA

Numerical data of physical quantities that we handle are typically gridded, whether regularly or not. A first step to handle the data concisely would be to combine them with their grid values and other information such as units to form an "object". This is the way of organizing data that file formats such as NetCDF and HDF4 employ to make data contents self-descriptive. It then becomes possible to devise abstract operations on the data as physical quantities rather than just as numerical arrays. An example of such operations is to slice a multi-dimensional data in terms of physical coordinate values. The organization into an object would also serve for visualization, since axes and titles can be drawn automatically from information stored in the object.

3. IMPACTS OF DATA-HANDLING CONSOLIDATION

By using object-oriented languages, we can hide internal structure of data objects from the user and make him or her access them only through abstract operations. The accessors can be the same in many cases whether the actual data resides entirely on computer memory or are kept in a file (letting the data object consisting of file handlers). Self-descriptive file formats such as those stated above can easily be adapted to this framework, and even non-self-descriptive formats can be conformed to it if the user provides ancillary information needed. It should be emphasized that distributed data whose entity consists of multiple files can also be adopted to this framework. The files may

possibly be distributed across network.

Since broad formats can be covered and the accessors to data will be consolidated as much as possible, users of the library would naturally develop applications that can be used easily by others. Therefore, the library is expected to become a basis on which data-handling applications are developed and shared in research communities.

4. IMPLEMENTATION

To realize such data handling we have been developing a class library for use in the object-oriented language Ruby. Ruby is perhaps the best object-oriented scripting language to date and is freely available from http://www.ruby-lang.org. Since it can be used interactively, it is suitable for interactive data analysis. Yet, an interactive trial and error can be organized smoothly into a program if needed. It is also noteworthy that functions written in the C Language is easily incorporated to Ruby so that existing C libraries (or Fortran libraries via f2c) can be utilized with small cost. Since Ruby offers strong network support, we are envisioning to extend our library to handle data across network. The original distribution of Ruby, however, does not have efficient multi-dimensional numerical arrays or a user-friendly scientific visualization tools needed by the library. The companion paper by Kawanabe et al (P1.20) presents our development of such infrastructure.

The development of the high-level library to handle physical data is now on a preliminary stage. We are supporting the NetCDF file format as the fist step and implementing mathematical and statistical operations as well as graphics to physical data objects stored self-descriptively in files. We will present its outline and demonstrate the preliminary version in the talk.

5. FUTURE DIRECTION

We are planning the following development.

- Comprehensive support of graphics, mathematical and statistical functions
- Array vectorization support for numerical simulation
- Support self-descriptive file formats such as HDF in addition to NetCDF
- Develop template forms for non- or imperfectly self-descriptive file formats
- Distributed file support
- File access over network made similarly to locally stored data