Ae-002 会場：C307 時間：6月26日 9:12-9:24

Ruby による DCL 拡張

DCL extension for Ruby: toward data objectization and efficient analysis

 # 後藤 謙太郎[1]，塩谷 雅人[2]，沼口 敦[3]，堀之内 武[4]，高橋 千賀子[5]，林 祥介[6]
Kentaro GOTO[1], Masato Shiotani[2], Atusi NUMAGUTI[3], Takeshi Horinouchi[4], Chikako Takahashi[5], Yoshi-Yuki Hayashi[6]

[1] 北大院・理・数，[2] 北大院・地球環境，[3] 北大・地球環境，[4] 京大・超高層，[5] 富士通エフ・アイ・ピー株式会社，[6] 北大・理・地球惑星
[1] Division of Math. Hokkaido Univ., [2] Graduate School of Environmental Earth Science, Hokkaido Univ., [3] EES, Hokkaido Univ., [4] Radio Atmos. Sci. Center, Kyoto Univ., [5] FUJITSU FIP CORPORATION, [6] Earth and Planetary Sci., Hokkaido Univ.

http://www.notwork.org/earth2000/

Ruby のための DCL 拡張を試みた．これは IDL のような多機能性と対話性を提供するもので，なおかつ，NetCDF のようなデータフォーマットを直接扱えるなど，拡張も容易である．結果として，柔軟かつ素早くデータ解析を行う開発環境が得られる．

1．概要

Ruby のための DCL 拡張を試みた．これは IDL のような多機能性と対話性を提供するもので，なおかつ，NetCDF のようなデータフォーマットを直接扱えるなど，拡張も容易である．結果として，柔軟かつ素早くデータ解析を行う開発環境が得られる．

2．IDL とは

IDL(Interactive Data Language)は，描画と数学計算ライブラリを備えた，対話的に操作できるインタプリタ言語である(米 RSI Inc.の商用ソフトウェア)．言語の骨格は Fortran に似ているが，変数の型は動的に決まる．データの種類としては，数値スカラ，文字列，配列，構造体をサポートする．現行の第 5 版からオブジェクト指向でいうクラスの定義が一応可能になったが，必ずしも使い良いとは言えない．

2.1 IDL でのデータ解析・可視化

IDL では，「物理量」の単位等の属性や「軸」(独立変数)を持つ構造化されたデータを扱うために，構造体を利用出来る．NetCDF のような自己記述性をもつデータフォーマットに対しては，ファイル上の変数を構造体で管理することで，作画等の属性や軸に関する情報を自動的に入手・操作するようなライブラリを構築可能である．そういったライブラリを作れば，快適にデータの操作と可視化を行うことも出来る．

2.2 IDL の問題点

上で「作れば」と書いたが，問題はここにある．IDL は拡張性・柔軟性に欠ける．例えば，部分配列を範囲で指定することは出来るが，これに読みとばしのステップを導入することは不可能である．つまり，仕様にないことをプログラミングで解消する自由度は小さい．言語の近代化も足りず，例えばガベージコレクション(GC)は存在しない．最後に，フリーでないということも挙げておく．かなり高価なことも問題であるが，ソースコードが公開されていないためユーザによる改良の自由がないことも障害となる．

3．Ruby

Ruby はまつもとゆきひろ氏が設計開発したオブジェクト指向スクリプティング言語であり，次のような特徴を持つ：オープンソースフリーソフトウェア，インタプリタ，すべてがオブジェクト，シンプルな文法，型を持たない変数，再定義可能な組み込みメソッド，引数に出来るクロージャ，GC，プラットフォームに依存しないスレッド，活発な国内のコミュニティなど．

3.1 Ruby の拡張ライブラリ

Ruby は Perl や Python のような他のスクリプティング言語と同様，既存ライブラリを利用する拡張ライブラリを容易に作成出来る．本研究においては，地球流体電脳倶楽部による Fotran ベースの基本関数・描画ライブラリである DCL を Ruby から使うための拡張を作成した．また Ruby の配列は任意の要素を格納できるが，自由度から来るオーバヘッドが我々の大規模計算では無視できないため，数値に特化した多次元配列を表現する NumArray クラスを作成した．これは多次元配列に必須の機能(部分配列の取り出し，指定された間隔による繰り返し処理など)を持たせたものである．

3.2 Ruby による利点

Ruby はインタプリタなので急いで書き下すことや．プロトタイピングに向いている．インタプリタ固有の遅さは時間のかかる処理を予め C や Fortan で書いて，実行時にリンクすれば補える．さらに柔軟な表現が可能である点が特筆に値する．例えば，3 次元配列 a の各要素に三角関数 sin を施した結果の第 1,3 成分を固定した断面の和を i=1..n の範囲でとることを，

```
sum(1,n){|i| sin(a)[0,i,3]}
```

と書くことも出来る．また GTK や TK も用意されているので，プラットフォームに依存しない GUI の作成は容易である．GNU の Readline も利用できるため，ユーザが自分のための対話環境を構築することもたやすい．

バイナリ形式を扱うためのメソッドも我々が一部拡張し，IEEE754 形式の浮動小数点数のバイナリフォーマットもエンディアンを考慮しつつ扱えるようになった．また近年発達した Yacc 相当のパーザ生成器を使えば文脈自由言語で書かれたファイルも解釈できるので，NetCDF の拡張等も手軽に試せる．これは既存データをオブジェクト化するための橋渡しとなることを意味する．

4. まとめ

Ruby のもつ柔軟性は以前より認められているが，数値計算に本格的に応用しようという動きはこれまで無かった．そのため現時点では少なくとも数値計算界での Ruby の認知度は決して高いとはいえない．しかし，記述の柔軟性やオープンソースのもとでの拡張のしやすさを考慮すれば，特に計算資源の低価格化が進む今日，開発効率の向上のための当アプローチは極めて有効であると考える．