集合および関係演算に基づくプログラミング言語Rの基本機構について

2D-1

内田智史・木村英一・間野浩太郎

青山学院大学理工学部経営工学科

1．はじめに

我々は、集合および関係演算に基づく汎用プログラミング言語Rを設計・開発中である。この言語では、全てのデータオブジェクトを集合物であるとみなし、プログラムの作成過程を、集合の要素間の関係演算として捕える。こうすることにより、プログラム上での手続きの順序をあまり考えなくても良くなる。手続きの順序、すなわち、タイミングの問題は、人間にとって非常に不得意な分野であり、プログラミング効率の向上、バグ発生率の低下が期待できる。要素間の関係として非手順的に表現されたソースプログラムは、その集合演算に対応する高速なアルゴリズムに変換される。

この言語の狙いは、

[1]プログラムの持つ動的な構造---特にオブジェクト指向の持つ非常に動的な構造を、その記述性を低下させずに静的な構造で表現する方法論の開発。

[2]アルゴリズムの自然な表現方法の研究。

[3]高度に抽象化されたプログラミング環境におけるデバッグ手法の開発。

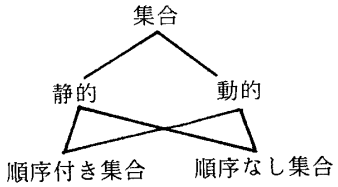
などである。また、Rはプログラムの記述に関して、単にアスキー文字列にこだわらず、プログラマの感覚に合せた自由な記号を用いるようにしている。従って、エディタを含めたRの支援ツールは、既存のものが使用できず、新たに構築しなければならない。それに関しては、次の発表(2d-2)で述べる。

2．R言語の概要

Rのプログラム構成要素は、(1)集合定義、(2)規則定義、(3)手順定義、である。プログラムで扱うデータは、すべて集合として定義される。規則とは、ある条件が発生した場合にとるべき行動を規定する。規則を適用するタイミングは、処理系が認識し決定する。集合定義、規則定義だけでは、完全にプログラムを記述できない場合がある。つまり、どうしても手順的な操作が必要になる場合である。このような場合は、手順定義で指定する。また、プログラムの全体的な流れの記述にも用いられる。

2.1 Rにおける集合

Rでは、集合を中心にしてプログラムを作成する。Rは、次のような集合の階層を持っている。



静的とは要素の構造がプログラムの実行中変化しないことを意味し、動的とは要素の構造がプログラムの実行中に変化することを示している。『順序付き集合』とは、要素の順序が意味を持つ集合であり[...]で表現し、『順序なし集合』とは、要素の順序が意味を持たない集合であり、{...}として表現する。

集合は、一般的に次のような記法で表現する。

集合オブジェクト名:種別名　集合記述

たとえば、

```
InKey:Command{Display Input Compute}
```

InKeyという順序なし集合は、Command種別に属し、その要素として、Display,Input,Computeがあることを示している。

2.2 プログラムの構成

Rでは、プログラム・関数などは、順序なし集合として定義する。その主な要素として、以下のものがある。

1) 引数:このプログラムに対する引数文字列を格納した順序付き集合
2) データ特性:このプログラムで使用するデータ構造を集合として定義する順序なし集合
3) 規則の記述:与えられた条件に対して取るべきアクションを記述する順序なし集合
4) 手順の記述:どうしても手順を取り去ることの出来ないアルゴリズムの記述を行う。

2.3 種別

集合は、任意の『種別(kind)』に属している。種別とは、オブジェクト指向におけるクラスに似ている。Rでは、組込み種別として、integer,stringなどのように基本的なデータタイプから、File,rs232cなどのような複雑な構造を持つ種別が用意されている。クラスと種別が大きく異なる点は、クラスでは、データ構造が完全に固定されており、また、その細部が隠蔽されメソッドを介さないとアクセスできないのに対し、種別は、基本的な構造のみが定義されており、その基本構造と矛盾を生じないかぎり、別の構造として再定義可能であるという点である。

たとえば、SequentialFile種別は、

```
[*c:char]
```

として、すなわち、文字の集合として定義されている。これは、

```
InFile:SequentialFile[*Line[LineBody:string,'¥n']]
```

すなわち、Lineという行（これは、文字列種別のLineBodyと改行記号からなる）から構成されるというようにプログラムの中で再定義してもかまわない。また、一般にその内部に対してアクセスすることができる。たとえば、i番めの要素は、要素名iでアクセスすることができる。

About The Principles of A Programming Language R Based on Set and Relation Operation

Satoshi Uchida, Eiichi Kimura, Koutaro Mano

AOYAMA GAKUIN UNIVERSITY

3．プログラム例

概念を少し明確にするために、プログラム例を示すことにする。プログラム例１に、wcプログラム(ファイル内の文字数、行数、単語数の表示)を示す。

```
wc:Program{Arg[FileNameStr:string];
   Def{ InFile:File(FileNameStr)
        = CharSet[*c:char]
        = LineSet[*Line[LineBody:string,'¥n']]
        = WordSet{*Word,*Separator};
        Separator:char{' ' '¥t' '¥n'};
        Word{*c:char!c not in Separator};
        Answer:数列[ !CharSet! !LineSet! !Word!]
       }
    Proc[ Answer →  Display ]
   }
```

プログラム例１：wcプログラム

Argは、引数集合であり、その第一要素に対象となるファイル名が格納されている。このプログラムでは、入力ファイルを３つの異なる構造としてDefで表現している。まず、最初(3行目)に、このファイルは、単なる文字の集合であることを示し、次(4行目)に改行記号('¥n')で区切られた行(Line)の集合であることを示し、最後(5行目)に単語と分離記号から成る集合であることを示している。そして、文字の集合の要素数が文字数に、行の集合の要素数が行数に、単語と分離記号の集合の内の単語の数が単語数に相当することを利用し、その答えから成る数列Answerを定義している。Procは、数列AnswerをDisplay(組込み記号でディスプレイ端末)に表示せよという意味である。

プログラム例２は、事務処理などで良く用いられるマッチングアルゴリズムをＲで記述した例を示している。このプログラムでは、入力の対象となっている２つのファイルを同じキーを持つものに対し、グループ化することによって、マッチング済みのファイルを定義している。

```
1次のマッチング:Program{
 Arg[3*FileName:string];
 Def{
    InFile1:SequentialFile(FileName1)
        [*Record[key1:string(5) data1:string '¥n']];
    InFile2:SequentialFile(FileName2)
        [*Record[Key2:string(5) data2:string '¥n']];
    OutFile:SequentialFile(FileName3)
        [*Record[key data1 data2]!
             (key=key1=key2)
          or (key=key1≠key2 data2=nil) ]
    }
 Rule{
    key=key2≠key1=>
    ErrorProc("トランザクションに誤りあり");
    }
 Proc[
      input[InFile1 InFile2];
      output[OutFile];
    ]
 ErrorProc(ErrorMessage:string)[
    Print(ErrorMessage);
    ]
}
```

プログラム例２　マッチング

4．処理系の概要

Ｒは、現在、設計・開発中であり、仕様の細部についてはまだ、決定されていない部分もある。具現化のポイントとなるところは、集合として記述された抽象度の高いプログラム記述をどのようにして具体的なアルゴリズムに変換するかということであろう。たとえば、プログラム例２のように記述されたプログラムを、一行ごとのファイル入出力を行うアルゴリズムに変換して効率良く実行できるようにする方法の開発である。全ての場合について、それを求める決定的な方法がある訳ではないので、その場合には、個々の『定石』を数多く収集して、知識工学的な手法によりその定石を、あてはめて解決することが考えられよう。

現在、開発中の処理系は、以下の２つである。

[1] Ｒインタプリタ：Ｒプログラムをそのままの形式で実行するインタプリタ。多くのデバッグ機能を含む。

[2] Ｒコンパイラ:Ｒプログラムを、80286用のオブジェクト指向機能を持つアセンブリ言語 snow に変換する。

共にμ-vax II上のCommon Lisp(Vax Lisp)で記述されており、インタプリタはμ-vax II上で、コンパイルされたオブジェクトコードは、PC98XA上で動作する予定である。

［参考文献］

1.内田智史、木村英一、間野浩太郎、他:オブジェクト指向に基づくアセンブリ言語プログラミング環境、情報処理学会第32回全国大会 6F-7,pp.453-454 (1986.3)
2.木村英一、内田智史、間野浩太郎:集合および関係演算に基づくプログラミング言語Ｒの支援環境について、情報処理学会第33回全国大会 2D-2,pp.375-376 (1986.10)