再帰プログラム作成のための会話型支援システムの検討と試作

2001MT105 土田 和宏
指導教員　真野 芳久

1　はじめに

通常、プログラム作成には、創造的な作業をともない、ある程度プログラム全体の処理の流れを把握する必要がある。しかし、繰り返しと同様に再帰も、問題によって複雑な動作を行う場合があり、プログラム実行時の状況をすべて把握してプログラミングをすることは困難である。これを軽減するためには、再帰構造を理解する方法を明確にし、プログラミングのための方法論を設定する必要がある。そして、その方法論に基づいた作成支援システムの開発が望まれる。

我々は、繰り返しに対する作成支援システムとして LiPLIS の設計・開発を進めている [1]。LiPLIS の特徴としては、ループ不変条件を利用して、複雑な繰返しの動作内容を理解する必要のないプログラミング支援を行っていることが挙げられる。一方、再帰に対しては、再帰プログラムの動作の理解を支援するシステムが研究されている [2]。しかし、この論文は再帰呼出しが行われるまでの実行系列の繰返しと再帰呼出しの復帰から関数の最後までの実行系列の繰返しをそれぞれ反復処理として考えており、プログラマは複雑な再帰の動作内容を理解する必要がある。

本研究では、再帰プログラムを支援対象とし、複雑な再帰の動作内容を理解する必要のないプログラミングのための方法論を設定し、その方法論に基づいた作成支援システムを検討していく。

2　再帰構造の理解方法

再帰プログラムを作成する上で重要なこととして、再帰構造を動的に理解するのか静的に理解するのかが挙げられる。[2] では、再帰プログラムの動作を理解させるために再帰構造を図を用いて表現し、さらにアニメーションを利用して再帰プログラムの動作の可視化を行っている。その特徴としては、

- 再帰呼出しを一つしか含まないような構造をもつ再帰プログラムを対象としている
- 再帰処理を二つの反復処理に対応させ、入力データから出力データまでの処理過程をアニメーションを用いて動的に表現している

の二点がある。

これは、再帰の動作をプログラマに理解させることに対しては有効であるが、再帰呼出しを一つしか含まない再帰プログラムを対象としている以上、数多くの再帰プログラムに対するプログラミング支援としては不十分なところがある。本研究では、数多くの再帰プログラムに対応できるようにするため、再帰を静的につまり、再帰関数の機能をその動作ではなく仕様によって理解する。

再帰構造を静的に理解することの特徴として、

- 再帰の部分問題の解が正しく得られると仮定し、その解を用いて元の再帰関数の解を求めることで、再帰関数を作成する。

が挙げられる (これを信念の再帰的飛躍 [3] という)。これにより、再帰の詳細な動作内容を理解する必要のないプログラミングができる。

3　プログラミングのための方法論

本研究では、再帰構造を静的に理解するという原則の下、以下の手順でプログラミングを行う方法論を考えた。

1. 問題の入力データの設定
2. 使用する再帰関数の役割 (入出力条件) の設定
3. 単純ケース時の条件と内容の設定
4. 部分問題の発見と再帰呼出し前後の処理の設定
   （a）部分問題の発見
   （b）再帰呼出しの前に実行する処理の設定
   （c）部分問題の引数の設定
   （d）再帰関数の解を求めるための処理の設定
5. 問題の出力条件を満たすための処理の設定

上述の方法論は [3] に記載されていた一般的な再帰プログラムの形式を参考にしたものであり、プログラマはこの方法論にもとづいてプログラミングをする。この方法論は再帰の部分問題の解が正しく得られると仮定することで、部分問題の詳細な動作内容を理解する必要がなく、動的に考えたプログラミングよりもプログラマの負担の軽減が期待できる。

4　プログラムの正しさ確認支援

実際にプログラムを作成したとしても、ソースコードを見ただけではそのプログラムが正しいかどうかは不明である。そこで、システムがプログラムの正しさの証明を手助けするための支援について検討していく。

通常、再帰プログラムの正しさを証明する際には、数学的帰納法を用いるが、システムが自らプログラムの正しさを証明することはできない。そのため、実際に再帰プログラムの正しさを証明するのはプログラマに委ね、本研究では、プログラマが作成したソースコードから正しさを証明するための手助けとなる情報を生成し、それを提示する支援について検討していく。

プログラマは仕様を満たすプログラムを作成する。したがって、仕様とプログラムは記述形式は異なるが、同

じ意味をもっていると言える。そこで、作成したソースコードを部分的に仕様の言葉を用いて変換し、それをプログラマに提示する。そして、プログラマは数学的帰納法により、プログラムの正しさを証明する。

数学的帰納法は、証明すべき命題に対し、「n = 1 の時」と「n ≦ k の時が正しいと仮定して n = k+1 の時」の 2 つの場合において命題が成り立つかを証明していく手法である。したがって、ソースコードを仕様の言葉を用いて部分的に変換したものは、数学的帰納法に利用できると考えられ、プログラムの正しさ確認支援の効果が期待できる。

5 再帰プログラミング支援システムの試作

5.1 システムの概要

再帰プログラムでは、必ず再帰呼出しや if 文を用いており、また、値を返す関数では、return 文もよく使用される。そのため、これらを変換の対象とすることで、プログラムの正しさの証明やプログラムの理解の手助けができるのではないかと考えた。そこで、方法論によって作成したソースコードをもとにして、それを適切な言葉に変換したものをプログラマに提示するシステムを考えた。下記にシステムの設計方針を示す。

1. 関数の仕様や変換される内容は、日本語とする。
2. 関数の仕様は、関数の引数をすべて使用したコメントとして記述する。
3. 変換の対象となる部分は、if 文と return 文、再帰呼出しの部分である。
4. 変換されなかったソースコードは、そのままプログラマに提示される。

5.2 システムの構成

本システムの働きとして、

- 再帰を静的に考えた上でのプログラミングにおける方法論を提示する
- プログラマが作成したソースコードをプログラムの正しさ確認に利用できるデータに変換する

の二つがあり、それぞれを方法論提示部、データ生成部とする。全体の流れの概略を図 1 に示す。

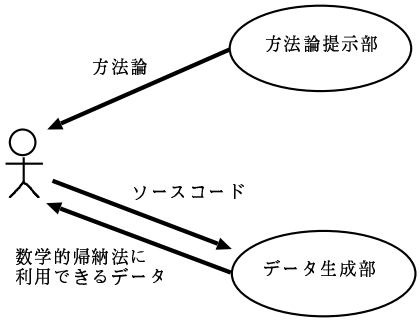

図 1 ユーザとシステムのやりとり

●方法論提示部　システムは再帰構造を静的に考えた方法論を提示する。プログラマはその方法論に基づいたプログラミングを行い、ソースコードを作成していく。

●データ生成部　作成されたソースコードの字句解析を行い、認識されたトークンによってそれぞれの処理を実行する。つまり、システムはソースコードを数学的帰納法に利用できるデータとして変換し、それをプログラマに提示する。再帰呼出しだけでなく、if 文や return 文も変換の対象とすることで、数学的帰納法による証明に必要な情報をより多く提供できる。

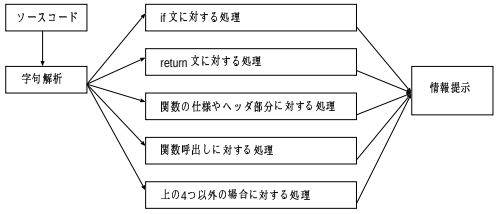

図 2 データ生成部における支援

図 2 の関数の仕様やヘッダ部分に対する処理において、この処理は再帰関数のみではなく、他の関数も処理対象とし、関数呼出しの部分は再帰関数と同じく、関数の仕様を用いて変換する (図 3)。

```
/* 非負整数 N の k 進数表現を出力する,
   ただし、2 ≦ k ≦ 10 */

void putRep(int N, int k){
  if (N < k)
    printf("%d",  N);
  else {
    putRep(N / k,  k);
    printf("%d", N%k);
  }
}
```

```
型 void
関数名 putRep
引数 int N, int k

N < k の時、
  printf("%d",  N);
それ以外の時、
  非負整数 N / k の k 進数表現を出力する,
  ただし、2 ≦ k ≦ 10
  printf("%d", N%k);
```

図 3 実行例

6 おわりに

本研究では、再帰構造を静的に考え、その考えをもとにプログラミングのための方法論を設定し、プログラムの正しさ確認のための支援について検討した。その結果、一般の再帰プログラムに適用できる支援システムを試作することができた。

ただ、コード作成において、まだユーザに依存するところが多く、よりプログラミング支援を行うための支援方法を検討することが今後の課題として挙げられる。

参考文献

[1] 古田ら：説明文書としてのループ不変条件を利用した文芸的プログラミング, 南山経営研究 (2002).

[2] 松田ら:プログラムの振舞いに基づく再帰プログラミングの教育支援, 電子情報通信学会論文誌, Vol.J80-D-II No.1, pp.326-335 (1997.1).

[3] Eric S.Roberts 著, 有川節夫, 篠原 武 訳：再帰的思考法, オーム社 (1993.4).