A-022

数式処理システム Mathematica 上における再帰除去システム
Recursion Removal System on Mathematica

市川 祐輔* 二村 良彦† 上田 和紀‡
Yusuke Ichikawa Yoshihiko Futamura Kazunori UEDA

1 はじめに

多項式漸化式は，様々な理数系の分野において用いられるため，Mathematica のような数式処理システム上においても利用されることが多い．しかし，これらの式は，再帰形式で記述されることが一般的である一方，その直接的な計算は重複計算および再帰呼び出しなどのオーバーヘッドが発生する．この点は，数を専門的に扱う数式処理システムにおいても同じ問題が生じる．実際，多項式漸化式は指数関数オーダーの再帰プログラムになることが多く，たとえ簡易な再帰プログラムであっても，実用的な範囲でプログラムの実行ができない．例えば，図 1 のプログラム F は，入力 100 以上では 10 分以内に計算ができない (表 1)．これらの現象は，再帰呼出によって生じる再帰の組み合わせ総数が指数関数オーダーであることを考慮すれば当然であるが，それでもこのような簡易なプログラムで再帰の深さが浅い場合にも実行が困難であることは，不便である．

そこで本稿では，Mathematica 上で多項式漸化式を線形オーダー反復プログラム (LINP) または対数関数オーダー反復プログラム (LOGP) に変換する再帰除去システム：R3F システムの実装を報告する．R3F システムは，再帰除去法である累積関数法 [2] に基づいており，Web 上においてダウンロードが可能である [3]．本システムにより，例えばプログラム F はプログラム LINP に変換される (図 1)．なお本稿では紙面の都合により省略したが，LOGP についても同様に変換可能である．R3F システムの変換により，変換前後でプログラムの計算時間の高速化が図られたことを，表 1 に示す．変換前後のプログラムの計算時間を比較すると，$x = 10$ については，3.766[ms] から 0.547[ms](LINP) 及び 0.469[ms](LOGP) とそれぞれ 6.5 倍及び 8.0 倍の高速化が得られた．また，$x = 100$ 以上については，10 分以内で計算されていないものが，変換後にはいずれも 10 分以内で計算可能となり，プログラムの高速化が達成できた．

なお，本稿では変換前のプログラムを原始プログラムと呼び，変換後のプログラムを剰余プログラムと呼ぶ．また，本稿における線形及び対数関数オーダーという名称は，プログラムの外形に着目した名称であり，それらの名称の示す計算量で計算できる場合もあるが，原始プログラムは外部プログラムを呼ぶこともできるなどの理由により，常にそれぞれの計算量で計算できるわけではない．特に，対数関数オーダー反復プログラムへの変換には，後述するように制御関数についての条件がある．

連立方程式に対しては，行列を用いた方法によっても計算することはできるため，同様な手法が多項式漸化式に対しても適用できる場合もある．しかし複雑な多項式漸化式に対しては，これらの変換は困難である．特に Mathematica のように初心者も使用するようなシステムにおいては，変換システムが有用である．

表 1 プログラム F の原始プログラムと剰余プログラム (LINP,LOGP) の計算実行時間の比較 [1000 回,msec]

入力	原始プログラム	LINP	LOGP
$x = 10$	3.766	0.547	0.469
$x = 100$	—	3.688	2.328
$x = 1000$	—	34.266	13.563
$x = 10000$	—	471.859	141.078

— は 10 分以上を示す．

計算環境：Windows XP, Pentium(R) 4 dual CPU 2.40 GHz, 512MB RAM.

```
F[x_]:=
 Which[x==0, 1, x==1, 0,x==2, 3
       ,x<=20, F1[x]+F[x-1]+2 F[x-2]+3 F[x-3]
       ,True,  5 F[x-1]+6 F[x-2]+7 F[x-3]];
F1[x_]:=If[x<=10,x+1,x-1];

CumLinpre[ x1_]:=
  Which[x1 == 0,1,x1 == 1,0,x1 == 2,3,
    x1 <= 20,CumLin[{{F1[x1]}, {1}, {2}, {3}}, x1],
    True,CumLin[{{0}, {5}, {6}, {7}}, x1]];
CumLin[vold_,  x1_]:=
 Module[{v=vold,vc,vv, u1, uu1,ter,tc={{0}}},
  vc={{1, 0, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}, {0, 0, 0, 0}};
  vv={{{1, 0, 0, 0}, {0, 0, 0, 0}, {3, 0, 0, 0}, {F1[u1], 1, 2, 3},
      {0, 5, 6, 7}}};
 u1=-1 + x1;
 While[True,ter=True; Which[
  u1 == 0, If[Not[tc[[1,1]]==1],tc[[1,1]]=1;vc=TRT[vc,vv[[1,1]],2]],
  u1 == 1, If[Not[tc[[1,1]]==2],tc[[1,1]]=2;vc=TRT[vc,vv[[1,2]],2]],
  u1 == 2, If[Not[tc[[1,1]]==3],tc[[1,1]]=3;vc=TRT[vc,vv[[1,3]],2]],
  u1 <= 20,tc={{0}}; vc=TRT[vc,vv[[1,4]],2]; ter=False, True,
If[Not[tc[[1,1]]==5],tc[[1,1]]=5;vc=TRT[vc,vv[[1,5]],2]];ter=False];
If[And[ter, And@@MapThread[And[Or[#1 == 0,#1 == 1,#1 == 2]]&,
                           {{u1, -1 + u1, -2 + u1}}]],Break[]];
   uu1=u1; v=vc . v; Clear[ uu1];u1=-1 + u1; ];
 Return[v[[1,1]] + PB1[u1] v[[2, 1]] +
          PB1[-1 + u1] v[[3, 1]] + PB1[-2 + u1] v[[4, 1]]] ];
PB1[ u1_]:=Which[u1 == 0, 1, u1 == 1, 0, u1 == 2, 3];
TRT[x_,y_,z_]:=Transpose[ReplacePart[Transpose[x],y,z]];
```

図 1 原始プログラム:F(上) 及び剰余プログラム:LINP(下)

* 早稲田大学大学院 理工学研究科 情報・ネットワーク専攻
† Futamura Institute, Inc.
‡ 早稲田大学理工学部 コンピュータ・ネットワーク工学科

2 組み込み関数 RSOLVE との比較

多項式漸化式を解く方法として，Mathematica の組み込み関数の RSOLVE があげられる．RSOLVE は漸化式の解を求める方法であるため，結果的に再帰を除去し，高速なプログラムを生成できる場合もある．しかし RSOLVE は R3F システムに比べて適用範囲が狭い．例えば述語関数について，RSOLVE は等号のみを対象とする．他方，R3F システムは任意の述語関数を対象としている．また，RSOLVE は変換中に浮動小数点を必要とする演算を導入する場合があるため，正確な計算ができない場合もある一方，R3F システムはそのような演算の導入はない．

3 使用方法

本節では，R3F システムの使用方法を述べる．まず，R3F システム [3] を Mathematica 上でロードし，以下の形式で原始プログラムなどの入力を行う．

$$\mathbf{R3F}[exp, fname, check]$$

ここで，*exp* が原始プログラムであり，*fname* は変換対象の関数名である．*check* は 1 であれば変換中に評価を行う．

3.1 原始プログラムのシンタックス

原始プログラムのシンタックスについて説明する．原始プログラムが N 個の再帰定義から構成されるとすると，*exp* は関数定義のリストであり，各関数は以下の形式の Mathematica 組込み関数:`Which` 関数である ($1 \le i \le N$):

$$f_i[x] = \mathtt{Which}[p_{i1}, Exp_{i1}, \ldots, p_{ij}, Exp_{ij}]$$

ここで x は複数の引数でもよい．p_{ij} を述語関数といい，Exp_{ij} は以下の形式の漸化式である:

$$c_{ij0}[x] + \sum_{r=1}^{N} \sum_{s=1}^{Fdeg} c_{ijrs}[x] f_r[d^s[x]]$$

ここで d は後継関数という．$Fdeg$ は原始プログラム内の最大の次数 (自然数) である．c は任意の有理数，関数呼出しまたは定数であり，制御関数という．全ての c が引数に依存していない場合には，対数関数オーダーで計算可能な剰余プログラムが導出される．なお原始プログラムに副作用は許されない．

3.2 後継関数

再帰呼び出しへの各引数の関数を，後継関数と呼ぶ．引数 x に後継関数が n 回適用されることを，$d^n(x)$ と書く．すなわち，$d^n(x) \equiv \overbrace{d(d(\ldots d}^{n}(x)\ldots))$．また，$n$ を次数という．後継関数は，以下の形式のいずれかであり，かつ 1) 後継関数は，再帰呼出の各位置において，同一の形式であること．2) 後継関数内で使用される変数名は，後継関数内の変数名の位置と関数の引数の位置が同一であること．3) 一つの再帰呼び出し内の後継関数の次数が同一であること，が満たされる必要がある．なお，C は自然数である．

$d(x)$	C の範囲
$x+C$	$1 \le C$
$x-C$	$1 \le C$
x/C	$C \le -1 \wedge 1 \le C$
$x*C$	$C \le -1 \wedge 1 \le C$

4 システムの構成

R3F システムは，累積関数法を，利便性を考慮しつつ実装したものである．

累積関数法は，累積関数定理及びその系によって構成されるプログラム最適化法である．その具体的な手順は，1) 対象関数の特定，2) 累積関数（累積行列）の導出，3) 累積関数定理の適用，である．また対数関数オーダー反復プログラムへの変換については，累積関数定理の系を適用することにより行われる．

ここでは，各ステップにおける実装の流れのみを述べる．1）においては，各種の関数（補助関数・述語関数・制御関数・後継関数）の特定をし，それらから次数を導出する．2）においては，次数・制御関数・述語関数から累積行列を生成する．なお，累積行列は理論的には複数生成されるが，これをそのまま実装すると組み合わせ総数の累積行列が生成され，不必要な累積行列も生成される場合があるため，実装においては一つの行列を生成しその行列要素の変換ルールを生成することにより，効率的なコード生成を行っている．3）累積行列・述語関数・後継関数に対して累積関数定理または系を適用することにより線形及び対数関数オーダーの反復の剰余プログラムが生成される．

ここで，累積関数定理とは，単一後継関数を持つ相互再帰プログラムから再帰除去を行うことを示す定理であり，その系とは，一定の条件の下，さらに対数関数オーダー反復プログラムに変換することを示す系である．詳細は論文 [2] を参照されたい．

5 まとめと今後の課題

本稿では，Mathematica 上において，多項式漸化式を線形オーダー反復プログラム (LINP) または対数関数オーダー反復プログラム (LOGP) に変換する R3F システムの実装を報告した．R3F システムは累積関数法 [2] に基づき変換しており，Web 上でダウンロードが可能である [3]．また，R3F システムによる変換前後の計算時間を比較した結果，例題においては少なくても 6.5 倍の高速化を達成したことを確認できた．今後の課題としては，剰余プログラムの最適化および累積関数法の完全な実装があげられる．

参考文献

[1] 二村良彦, 大谷啓記: 線形再帰プログラムからの再帰除去とその実際的効果, コンピュータソフトウェア, Vol. 15, No. 3, pp. 38–49 (1998).

[2] Y. Ichikawa, Z. Konishi, and Y. Futamura, “Recursion Removal from Recursive Programs with Only One Descent Function” *IEICE Trans. Inf.& Syst.*, vol.E88-D No.2, Feb., 2005.

[3] R3F SYSTEM: http://www.ueda.info.waseda.ac.jp/ ichikawa /r3f-index.html