プログラミング 6－14
（1996．3．27）

関数プログラムのプロモーション変換のための二手法の関係

岩崎 英哉 †　　胡 振江 ‡

†東京大学 教育用計算機センター
東京都文京区弥生 2-11-16

‡東京大学大学院 工学系研究科
東京都文京区本郷 7-3-1

関数プログラムでは，関数の間で受け渡されるが最終結果には出現しない中間的なデータ構造を生成しないように，関数合成をひとつの関数にプロモーションする (融合する) ことが，効率改善にとって重要である．補助引数あるいは蓄積引数を持つ関数に関してこの問題を解決するための手法として，“高階 catamorphism” を用いる方法と，“媒介型” を導入することによる hylomorphism を用いる方法が研究されている．本稿は，補助引数を持つ関数の高階 catamorphism による融合操作は，媒介型を用いた変換操作の枠組で説明することができることを，簡単な具体例を通して示す．

Relationship between two Approaches to the Promotional Transformation of Functional Programs

Hideya Iwasaki †　　Zhenjiang Hu ‡

†Educational Computer Centre, University of Tokyo
2-11-16 Yayoi, Bunkyo-ku, Tokyo 113, Japan

‡Faculty of Engineering, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan

In order to improve the efficiency of functional programs, it is important to perform promotional transformation to reduce intermediate data structures which are passed through a composition of two functions but never appear in the final result. As for the functions with accumulational parameters, two approaches for this transformation have been studied: one is based on the concept of “higher order catamorphism,” and the other is based on the derivation of hylomorphisms structured by the “medio type.” In this paper, through a concrete simple example, we show that the former approach can be totally interpreted within the framework of the latter.

1. はじめに

関数プログラムでは，基本的な機能を持つ小さな部品となる関数を関数合成によって組み合わせて，大きなプログラムを記述する．関数間で受け渡される中間的なデータ構造を生成しないように，関数合成をひとつの関数に融合する (プロモーションする) ことが，関数プログラムの効率改善にとって重要である．最近，構成的アルゴリズム論 [3] の立場からこの問題を解決していこうという研究が盛んに行なわれている．構成的アルゴリズム論では，catamorphism，anamorphism，hylomorphism などと呼ばれる範疇に属す関数が中心的な役割を果たす．

部品となる基本関数には，補助引数あるいは蓄積引数を持つものも多数ある．これを構成的アルゴリズム論で扱うための考え方に "高階 catamorphism" [1,4] がある．一方，単純な catamorphism では表現できないような関数を扱うための技法として "媒介型" [2] を用いた変換も研究されている．

本稿の目的は，「補助引数を持つ関数の高階 catamorphism による融合操作は，媒介型を用いた変換操作の枠組で説明することができる」ことを示す点にある．ただしページ数の制限から，一般的な場合についてこれを示すのではなく，本稿では *isum* という簡単な関数を具体例として両手法の変換手順を示し，お互いの関係を明らかにする．*isum* は，与えられたリストの先頭から各要素までの和をリストとして返す関数であり，その定義は以下で与えられる．

$$isum\,[\,]\,d = [d]$$
$$isum\,(x : xs)\,d = d : isum\,xs\,(x + d)$$

ここで d は補助引数であり，初期値としては一般的には 0 が与えらえる．たとえば，

$$isum\,[4,1,-2,3]\,0 = [0,4,5,3,6]$$

となる．$isum\,xs\,d$ の結果に "各要素に c を加える関数" $h = ((c+)*)$ を適用した式 $(c+)*(isum\,xs\,d)$ の融合変換を考える．ここで，c は適当な定数とする．うまく融合変換できれば，結果は $isum\,xs\,(c+d)$ となることが期待されるであろう．

2. 構成的アルゴリズム論

2.1 catamorphism

catamorphism とは，再帰的に定められた型を持つデータ上において "自然に" 再帰定義された関数をさす．たとえば，要素の型が a のリスト型は，

$$\mathsf{L}\,a = [\,] \mid \grave{a} : \mathsf{L}\,a$$

と定義される．ここで [] と : は型構成子である．この上に定義される自然な再帰関数 f は，$\oplus$ を適当な二項演算子として以下のように定義されるであろう．

$$f\,[\,] = e\,()$$
$$f\,(x : xs) = x \oplus (f\,xs)$$

ただしここでは，定数 (f [] の値) を「() を引数とする関数」を用いて表現した．この場合，e と $\oplus$ を定めれば f は一意に定まるので，この f (リスト上の catamorphism) のことを $(\!|e \triangledown \oplus|\!)_{\mathsf{L}}$ と書く．(添字は以下に述べる "リスト型を定義する関手" である．)

一般的には，型定義とその型のデータに対する操作は，関手 (functor) によって特徴付けられる．本稿においては，関手は以下の四つの基本操作から構成される．ここで X，Y は型を，f，g は関数を表すものとする．また，id は恒等関数である．

1. (恒等) $\mathsf{I}\,X = X$，$\mathsf{I}\,f = f$
2. (定数) $!a\,X = a$，$!a\,f = id$
3. (積) $X \times Y = \{(x,y) \mid x \in X, y \in Y\}$
   $(f \times g)\,(x,y) = (f\,x, g\,y)$
4. (直和) $X + Y = \{1\} \times X \cup \{2\} \times Y$
   $(f+g)\,(1,x) = (1, f\,x)$
   $(f+g)\,(2,x) = (2, g\,x)$
   $(f \triangledown g)\,(1,x) = f\,x$, $(f \triangledown g)\,(2,x) = g\,x$

上記のリスト型の場合の関手 L は $\mathsf{L} = !1 + !a \times \mathsf{I}$ となる．以下では，型名とその型に対応する関手を，同じ記号を使って表わすこととする．

型 F 上で定義される catamorphism $(\!|\phi|\!)_{\mathsf{F}}$ は，

$$(\!|\phi|\!)_{\mathsf{F}} \circ in_{\mathsf{F}} = \phi \circ \mathsf{F}\,(\!|\phi|\!)_{\mathsf{F}}$$

という式によって特徴付けられる．ここで in_{F} は関手 F で定義される型の構成子であり，リスト型の場合には $in_{\mathsf{L}} = [\,] \triangledown :$ となる．

2.2 融合定理

catamorphism におけるプログラム変換の基本となるのが，以下の融合定理 [3] である．

定理 1 (融合定理) 与えられた $(\!|\phi|\!)_{\mathsf{F}}$ と h に対して，

$$h \circ \phi = \psi \circ \mathsf{F}\,h \qquad \text{(A)}$$

を満足する ψ が存在すれば，$h \circ (\!|\phi|\!)_{\mathsf{F}}$ は $(\!|\psi|\!)_{\mathsf{F}}$ で表現されるような catamorphism となる． ∎

この定理の条件 (A) は，リストの場合には以下のふたつを満たすような e'，$\otimes$ の存在となる．

$$h\,(e\,()) = e'\,(), \quad h\,(x \oplus r) = x \otimes (h\,r)$$

この定理は，ある関数と catamorphism との合成が別の catamorphism で表現されるための条件を示している．この定理を適用することによって，ふたつの関数合成をひとつに融合することができ，

- ふたつの関数間で受け渡されるが，最終結果にはあらわれないような中間データ構造を除去することができる；
- 中間データ構造を辿る手間を省くことができる；

といった効果が期待される．

2.3 anamorphism と hylomorphism

catamorphism は，引数として与えられた再帰的なデータ構造を消費する関数であるが，これと "双

対" の関係にある関数，すなわち型 F のデータを再帰的に生成する関数を考えることができる．これを anamorphism [3] と呼び，$(\!|\psi|\!)_{\mathsf{F}}$ という記法を用いる．型 F 上で定義される anamorphism $(\!|\psi|\!)$ は，

$$out_{\mathsf{F}} \circ (\!|\psi|\!)_{\mathsf{F}} = \mathsf{F}\,(\!|\psi|\!)_{\mathsf{F}} \circ \psi$$

という式によって特徴付けられる．ここで，out_{F} はデータの分解子 (in_{F} の逆関数) であり，リストの場合は以下のようになる．

$$out_{\mathsf{L}}\ \mathsf{xs} = \mathsf{case\ xs\ of}\ [\,] \to (1,())$$
$$(\mathsf{x} : \mathsf{xs}) \to (2,(\mathsf{x},\mathsf{xs}))$$

たとえば，与えられた引数 n に対して [n, n − 1, ..., 2, 1] を返す関数 downto1 は，

$$\psi\,\mathsf{n} = \mathsf{case\ n\ of}\ 0 \to (1,())$$
$$\mathsf{m}+1 \to (2,(\mathsf{m}+1,\mathsf{m}))$$

を用いて $\mathsf{downto1} = (\!|\psi|\!)_{\mathsf{L}}$ と表現できる．

さらに，catamorphism $(\!|\phi|\!)_{\mathsf{F}}$ と anamorphism $(\!|\psi|\!)_{\mathsf{F}}$ との合成 $(\!|\phi|\!)_{\mathsf{F}} \circ (\!|\psi|\!)_{\mathsf{F}}$ を hylomorphism と呼び，$[\![\phi,\psi]\!]_{\mathsf{F}}$ という記法 [3] を用いることもある．hylomorphism には，η を F から G への自然変換とするとき，Hylo Shift という以下の性質がある．

$$[\![\phi \circ \eta, \psi]\!]_{\mathsf{F}} = [\![\phi, \eta \circ \psi]\!]_{\mathsf{G}}$$

この性質は，"二つ組" の hylomorphism の表現には自然変換分の自由度があることを示している．そこで本稿では，自然変換をくくり出した "三つ組" の hylomorphism の表現 [4,5] を用いる．三つ組表現によると，上の $[\![\phi \circ \eta, \psi]\!]_{\mathsf{F}}$ (あるいは $[\![\phi, \eta \circ \psi]\!]_{\mathsf{G}}$) は，$[\![\phi,\eta,\psi]\!]_{\mathsf{G,F}}$ と表わされる．三つ組表現における Hylo Shift は以下の通り．

$$[\![\phi \circ \eta, \mathsf{id}, \psi]\!]_{\mathsf{F,F}} = [\![\phi,\eta,\psi]\!]_{\mathsf{G,F}} = [\![\phi, \mathsf{id}, \eta \circ \psi]\!]_{\mathsf{G,G}}$$

hylomorphism はその一部に catamorphism と anamorphism を含むので，catamorphism における融合定理に相当するものと，その双対定理 (anamorphism における融合定理) [4,5] が成立する．

定理 2 (Hylo 融合定理)

$$h \circ \phi = \phi' \circ \mathsf{G}\,h \implies h \circ [\![\phi,\eta,\psi]\!]_{\mathsf{G,F}} = [\![\phi',\eta,\psi]\!]_{\mathsf{G,F}}$$
$$\psi \circ g = \mathsf{F}\,g \circ \psi' \implies [\![\phi,\eta,\psi]\!]_{\mathsf{G,F}} \circ g = [\![\phi,\eta,\psi']\!]_{\mathsf{G,F}} \quad \blacksquare$$

3. 高階 catamorphism による変換

以上は一引数関数に対する議論であったが，二引数以上の関数に関しては "引数をひとつもらい，関数を返す高階関数" としてとらえれば，通常の catamorphism の枠組で説明することもできる．このような catamorphism を "高階 catamorphism" [1] と呼ぶ．言いかえれば，高階 catamorphism とは，引数をひとつ適用した結果がまた関数となっているような catamorphism をいう．

3.1 高階融合定理

高階 catamorphism に関する融合定理 [1] は，以下のようになる．$h\,((\!|\phi|\!)_{\mathsf{F}}\ x\ a)$ という式で補助引数の a を取り除いて $(h\circ)$ とセクション化すると $(h\circ)\,((\!|\phi|\!)_{\mathsf{F}}\ x)$ と変形できるので，通常の融合定理の h が $(h\circ)$ に対応することがわかる．よって以下が得られる．

系 3 (高階融合定理 1)

$$(h\circ)\circ\phi = \psi \circ \mathsf{F}\,(h\circ) \quad \text{(B)}$$
$$\implies (h\circ)\circ(\!|\phi|\!)_{\mathsf{F}} = (\!|\psi|\!)_{\mathsf{F}}$$
すなわち $\quad h\,((\!|\phi|\!)_{\mathsf{F}}\ x\ a) = (\!|\psi|\!)_{\mathsf{F}}\ x\ a \quad \blacksquare$

リストの場合には，適当な e'，$\otimes$ が存在して，

$$h\circ(e\,()) = e'\,(), \quad h\circ(x \oplus r) = x \otimes (h\circ r)$$

を満たすことが条件 (B) となる．

ところが実際にこの定理を適用しただけでは，つまらない結果に終ってしまう場合が多い．その理由は，この定理だけでは補助引数に対する "操作" を加えることができないからである．したがって，融合結果の catamorphism から関数を抽出して補助引数に適用させるような，もうひとつの融合定理が必要である．

系 4 (高階融合定理 2)

$$(\circ g)\circ\phi = \psi \circ \mathsf{F}\,(\circ g) \quad \text{(B')}$$
$$\implies (\circ g)\circ(\!|\phi|\!)_{\mathsf{F}} = (\!|\psi|\!)_{\mathsf{F}}$$
すなわち $\quad (\!|\phi|\!)_{\mathsf{F}}\ x\ (g\ a) = (\!|\psi|\!)_{\mathsf{F}}\ x\ a \quad \blacksquare$

上の式を右辺から左辺に向かって読むと，高階 catamorphism から補助引数 の部分へ関数 (g) を取り出すための規則として解釈することができる．

リストの場合には，適当な e''，$\ominus$ が存在して，以下の式を見たすことが条件 (B') となる．

$$(e'\,())\circ g = e''\,(), \quad x \otimes (r\circ g) = (x \ominus r)\circ g$$

以上をまとめると，ある関数と高階 catamorphism の合成は，まず "高階融合定理 1" で高階 catamorphism に変換し，さらに "高階融合定理 2" で補助引数部に関数を抽出することによって，効率のよいプログラムに変換されることになる．

3.2 *isum* の変換

以後，高階 catamorphism による変換には *isum* といった字体を用いる．高階 catamorphism を用いた $h\,(isum\ xs\ d)$ の変換は，以下のような手順で進む．

1. *isum* を高階 catamorphism で表現する．その結果，$h\,((\!|p|\!)_{\mathsf{L}}\ xs\ d)$ という式になる．
2. h を融合することにより，$(\!|p'|\!)_{\mathsf{L}}\ xs\ d$ となる．
3. 関数 g を抽出して，$(\!|p''|\!)_{\mathsf{L}}\ xs\ (g\ d)$ とする．

3.2.1 *isum* の高階 catamorphism による表現

$$isum\,[\,] = \lambda d\,.\,[d]$$
$$isum\,(x : xs) = \lambda d\,.\,d : isum\,xs\,(x+d)$$

なので，

$$p_1\,() = \lambda d\,.\,[d]$$
$$p_2\,(x,r) = \lambda d\,.\,d : r\,(x+d)$$

となり，$p = p_1 \triangledown p_2$ とおくと $isum = (\!|p|\!)_{\mathsf{L}}$.

3.2.2 h の融合

高階融合定理 1 より，$(h\circ)\circ p = p' \circ \mathsf{L}\,(h\circ)$ となる p' を探せばよい．

まずはじめに，$(h\circ)\circ p_1 = p_1'$，すなわち，

$$((h\circ)\circ p_1)\,()\,d = p_1'\,()\,d$$

となる p_1' をみつける．

$$\begin{aligned}\text{左辺} &= h\,(p_1\,()\,d)\\ &= [c+d]\end{aligned}$$

なので，以下のように定めればよい．

$$p_1'\,() = \lambda d\,.\,[c+d]$$

次に，$(h\circ)\circ p_2 = p_2' \circ (id \times (h\circ))$，すなわち，

$$((h\circ)\circ p_2)\,(x,r)\,d = p_2'\,(x, h\circ r)\,d$$

となる p_2' をみつける．

$$\begin{aligned}\text{左辺} &= h\,(p_2\,(x,r)\,d)\\ &= (c+)*(d : r\,(x+d))\\ &= (c+d) : ((c+)*(r\,(x+d)))\\ &= (c+d) : ((h\circ r)\,(x+d))\\ &= p_2'\,(x, h\circ r)\,d\end{aligned}$$

よって p_2' は次のようにすればよい．

$$p_2'\,(x,r) = \lambda d\,.\,(c+d) : r\,(x+d)$$

以上より $p' = p_1' \triangledown p_2'$ とおくと $h\,((\!|p|\!)_{\mathsf{L}}\,xs\,d) = (\!|p'|\!)_{\mathsf{L}}\,xs\,d$ と変換された．

3.3 g の抽出

$(\circ g)\circ p'' = p' \circ \mathsf{L}\,(\circ g)$ となる p'' と g を探す．

まず，$(\circ g)\circ p_1'' = p_1'$，すなわち，

$$((\circ g)\circ p_1'')\,()\,d = p_1'\,()\,d$$

となる p_1'' をみつける．

$$\begin{aligned}\text{左辺} &= ((\circ g)\,(p_1''\,()))\,d\\ &= ((p_1''\,())\circ g)\,d\\ &= p_1''\,()\,(g\,d)\\ \text{右辺} &= [c+d]\end{aligned}$$

よって，$g\,x = c + x$ と定義すると以下になる．

$$p_1''\,() = \lambda d\,.\,[d]$$

次に，$(\circ g)\circ p_2'' = p_2' \circ (id \times (\circ g))$，すなわち，

$$((\circ g)\circ p_2'')\,(x,r)\,d = (p_2' \circ (id \times (\circ g)))\,(x,r)\,d$$

となる p_2' をみつける．

$$\begin{aligned}\text{左辺} &= p_2''\,(x,r)\,(g\,d)\\ &= p_2''\,(x,r)\,(c+d)\\ \text{右辺} &= p_2'\,((id \times (\circ g))\,(x,r))\,d\\ &= p_2'\,(x, r\circ g)\,d\\ &= (c+d) : (r\circ g)\,(x+d)\\ &= (c+d) : r\,(x+c+d)\end{aligned}$$

よって，以下の式が導かれる．

$$p_2''\,(x,r) = \lambda d\,.\,d : r\,(x+d)$$

$p'' = p_1'' \triangledown p_2''$ として以上をまとめると，

$$(c+)*(isum\,xs\,d) = (\!|p''|\!)_{\mathsf{L}}\,xs\,(c+d)$$

となる．ここで，実は $p_1'' = p_1$，$p_2'' = p_2$ なので $isum'' = (\!|p''|\!)_{\mathsf{L}}$ は $isum = (\!|p|\!)_{\mathsf{L}}$ と等しくなる．結局，

$$(c+)*(isum\,xs\,d) = isum\,xs\,(c+d)$$

となり，期待する結果を導出することができた．

4. 媒介型による変換

4.1 媒介型の導入

本節で述べる変換 [2] では，複数の引数をカリー化せず組として一体化し，単一引数とする．たとえば，isum の定義を以下のようにする．

```
isum ([], d) = [d]
isum (x : xs, d) = d : isum (xs, x + d)
```

しかし (一体化された) 引数は単純に再帰定義された型にはならないので，このままでは catamorphism や hylomorphism のような枠組にはおさまらない．そこで，一体化された引数を，単純に再帰定義された型へと "型変換" することを考える．

具体的には，注目する関数 f :: F × G → T に対して，適当な型 M ("媒介型" と呼ぶ) を考え，f を f' :: M → T と t :: F × G → M との合成 f = f' ∘ t として表現する．ここで，t は f の引数 (x, y) を M 型に変換するような型変換関数である．f' がうまく M 上の catamorphism $(\!|\phi|\!)_{\mathsf{M}}$ として表現できれば，融合定理等を利用して f を含むプログラムを構成的に変換していくことが可能となる．

以上の媒介型を用いた変換には，以下の性質 [2] があることが示されている．

- 型変換関数 t は 実は anamorphism として表現することができる．
- 中間にあらわれる媒介型 M は，変換の途中にのみ出現して最終結果では解消する．

前者の性質より，$\mathsf{t} = [\![\psi]\!]_{\mathsf{M}}$ とおくと，f は

$$\mathsf{f} = [\![\phi, \mathsf{id}, \psi]\!]_{\mathsf{M},\mathsf{M}}$$

と表現される．

対象とする関数が hylomorphism で表現されれば，あとは hylomorphism の融合定理を用いて変換していけばよい．その際，自然変換が hylomorphism の内部を自由に移動できる性質 (Hylo Shift) を利用し，hylomorphism の左側から関数が合成されている場合には自然変換を左側 (ϕ 側) に寄せ，hylomorphism の右側から関数が合成されている場合には自然変換を右側 (ψ 側) に寄せて，融合定理を適用しやすくすることも必要である．このように

hylomorphism の内側で自然変換を移動することを“hylomorphism の再構成”と呼ぶ．

与えられた f の定義から，媒介型と hylomorphism を導出するアルゴリズム，hylomorphism の再構成アルゴリズムの詳細は，文献 [2] を参照されたい．

4.2 isum の変換

以後，媒介型を用いた変換には isum という字体を用いる．h ∘ isum の変換の手順は，以下のようになる．ここで，$h = ((c+)*)$ である．

1. isum を hylomorphism で表現する．その結果，上式は $h \circ [\![p, id, q]\!]_{M,M}$ となる．ここで M は，導入された媒介型とする．
2. h を融合して $[\![p', id, q]\!]_{M,M}$ とする．
3. $p' = p'' \circ n$ となるような自然変換 $n :: M \to M'$ を p' から抽出する．さらに n を hylomorphism の中で移動すると，$[\![p'', id, n \circ q]\!]_{M',M'}$ となる．
4. 関数 g を anamorphism 側 (右側) から抽出して，$[\![p'', id, q'']\!]_{M',M'} \circ g$ とする．

以上 1～5 より，次式が導かれる．

$$h\ (isum\ (xs, d)) = [\![p'', id, q'']\!]_{M',M'}\ (g\ (xs, d))$$

4.2.1 isum の hylomorphism による表現

$isum = isum' \circ t$ と変形する．ここで $t :: [Int] \times Int \to M$ は型変換関数であり，$isum'$ は M 上の catamorphism である．媒介型 M は D_1，D_2 を型構成子として，

$$M = D_1\ Int \mid D_2\ (Int, M)$$

と定義され，この型を定める関手 M は $M\phi = id + id \times \phi$ となる．t の定義は，以下で与えられる．

$$t\ ([], d) = D_1\ d$$
$$t\ (x : xs, d) = D_2\ (d, t\ (xs, x + d))$$

実はこれは M 上の anamorphism である．実際，

$$q\ (xs, d) = \text{case } xs \text{ of}$$
$$[] \to (1, d)$$
$$(x : xs) \to (2, (d, (xs, x + d)))$$

とすると，$t = (\!|q|\!)_M$ と表現できる．一方，$isum' :: M \to [Int]$ は以下のような catamorphism となる．

$$isum'\ (D_1\ d) = [d]$$
$$isum'\ (D_2\ (d, r)) = d : isum'\ r$$

$p = p_1 \triangledown p_2$ とおいて $isum' = (\!|p|\!)_M$ とすると，

$$p_1\ d = [d]$$
$$p_2\ (d, r) = d : r$$

となる．上を総合すると，$isum = [\![p, id, q]\!]_{M,M}$ という hylomorphism で表現できたことになる．

4.2.2 h の融合

hylomorphism の融合定理より，$h \circ p = p' \circ M\ h$ となる p' を発見すればよい．

最初に，$h \circ p_1 = p'_1$ すなわち

$$(c+) * (p_1\ d) = p'_1\ d$$

となる p'_1 をみつける．

$$左辺 = [c + d]$$

なので，$p'_1\ d = [c + d]$ と定めることができる．

次に，$h \circ p_2 = p'_2 \circ (id \times h)$ すなわち

$$(c+) * (p_2\ (d, r)) = p'_2\ (d, (c+) * r)$$

となる p'_2 を探す．

$$左辺 = (c+) * (d : r)$$
$$= (c + d) : ((c+) * r)$$

なので，$p'_2\ (d, r) = (c + d) : r$ とおけばよい．

$p' = p'_1 \triangledown p'_2$ とおくと，以上の議論から

$$h \circ [\![p, id, q]\!]_{M,M} = [\![p', id, q]\!]_{M,M}$$

と融合された．

4.2.3 hylomorphism の再構成

p' から自然変換 n を抽出して，$p' = p'' \circ n$ とする．このためには，p'_i の右辺から，なるべく大きな“非再帰部”を自然変換中に抽出する必要がある．

$p'_1\ d = [c + d]$ については右辺全体が非再帰部であるから，以下のように p''_1，n_1 を定める．

$$p''_1\ u = u, \quad n_1\ d = [c + d]$$

次に，$p'_2\ (d, r) = (c + d) : r$ の右辺では $(c + d)$ が非再帰部なので，以下のように定める．

$$p''_2\ (u, r) = u : r, \quad n_2\ (d, r) = (c + d, r)$$

よって，$p'' = p''_1 \triangledown p''_2$，$n = n_1 + n_2$ とおけば，p' は $p'' \circ n$ と分解される．n を M から M' への自然変換とすると，M' は D'_1，D'_2 を型構成子として

$$M' = D'_1\ [Int] \mid D'_2\ (Int, M')$$

と定義され，この型をを定める関手 M' は $M'\phi = id + id \times \phi$ となる．最後に n を hylomorphism 内を右に移動して $q' = n \circ q$ とおくことによって，次式を導くことができる．

$$[\![p', id, q]\!]_{M,M} = [\![p'', id, q']\!]_{M',M'}$$

4.2.4 g の抽出

$q' = n \circ q$ に従って q' を書き下すと以下の通り．

$$q'\ (xs, d) = \text{case } xs \text{ of}$$
$$[] \to (1, [c + d])$$
$$(x : xs) \to (2, (c + d, (xs, x + d)))$$

ここで，$M'\ g \circ q' = q'' \circ g$ すなわち

$$(M'\ g \circ q')\ (xs, d) = (q'' \circ g)\ (xs, d)$$

となる g，q'' をみつけたい．

$$(M'\ g \circ q')\ (xs, d)$$
$$= ((id + id \times g) \circ q')\ (xs, d)$$
$$= \text{case } xs \text{ of}$$
$$[] \to (1, [c + d])$$
$$(x : xs) \to (2, (c + d, g\ (xs, x + d)))$$

であるから，$g\ (xs, d) = (xs, c + d)$ と定義すれば，

表 1: 高階 catamorphism による変換と媒介型による変換との対応

高階 catamorphism による変換		媒介型による変換	
isum の定義	$isum\,(x:xs) =$ $\quad \lambda d\,.\,\mathcal{E}\,(d,\ isum\,xs\,(x+d))$ $\mathcal{E}\,(d,r) = d:r$	isum (x : xs, d) = $\quad \mathcal{E}$ (d, isum (xs, x + d))	isum の定義
catamorphism	$isum\,xs\,d = (\!\lvert p \rvert\!)_{\mathsf{L}}\ xs\,d$ $p_2\,(x,r) = \lambda d\,.\,\mathcal{E}\,(d,\ r\,(x+d))$	isum (xs, d) = $[\![$ p, id, q $]\!]_{\mathsf{M,M}}$ (xs, d) p_2 (d, r) = $\mathcal{E}$ (d, r) q (x : xs, d) = (2, (d, (xs, x + d)))	hylomorphism
h の融合	$h\,((\!\lvert p \rvert\!)_{\mathsf{L}}\ xs\,d) = (\!\lvert p' \rvert\!)_{\mathsf{L}}\ xs\,d$ $p'_2\,(x,r) = \lambda d\,.\,\mathcal{E}'\,(d,\ r\,(x+d))$ $\mathcal{E}'\,(d,r) = (c+d):r$	h ($[\![$ p, id, q $]\!]_{\mathsf{M,M}}$ (xs, d)) = $\quad [\![$ p′, id, q $]\!]_{\mathsf{M,M}}$ (xs, d) p'_2 (d, r) = $\mathcal{E}'$ (d, r)	h の融合
		$[\![$ p′, id, q $]\!]_{\mathsf{M,M}} = [\![$ p″, id, q′ $]\!]_{\mathsf{M',M'}}$ p''_2 (u, r) = $\mathcal{E}''_i$ (u, r) q′ (x : xs, d) = (2, (c + d, (xs, x + d)))	再構成
g の抽出	$(\!\lvert p' \rvert\!)_{\mathsf{L}}\ xs\,d = (\!\lvert p'' \rvert\!)_{\mathsf{L}}\ xs\,(g\,d)$ $p''_2\,(x,r) = \lambda d\,.\,\mathcal{E}''\,(d,\ r\,(x+d))$ $\mathcal{E}''\,(u,r) = u:r$ $g\,d = c+d$	$[\![$ p″, id, q′ $]\!]_{\mathsf{M',M'}}$ (x, d) = $\quad [\![$ p″, id, q″ $]\!]_{\mathsf{M',M'}}$ (g (x, d)) p''_2 (u, r) = $\mathcal{E}''$ (u, r) q″ (x : xs, d) = (2, (d, (xs, x + d))) g (x, d) = (x, g d)	g の抽出
$isum''$ の定義	$isum''\,(x:xs) =$ $\quad \lambda d\,.\,\mathcal{E}''\,(d,\ isum''\,xs\,(x+d))$	isum″ (x : xs, d) = $\quad \mathcal{E}''$ (d, isum (xs, x + d))	isum″ の定義

上式 = case xs of
[] → (1, [c + d])
(x : xs) → (2, (c + d, (xs, x + c + d)))

となり，これが (q″ ∘ g) (xs, d) = q″ (xs, c + d) と等しくならなければならない．よって q″ は以下のように定義すればよい．

q″ (xs, d) = case xs of
[] → (1, [d])
(x : xs) → (2, (d, (xs, x + d)))

以上すべてをまとめると，

$$[\![\mathsf{p''},\mathsf{id},\mathsf{q'}]\!]_{\mathsf{M',M'}} = [\![\mathsf{p''},\mathsf{id},\mathsf{q''}]\!]_{\mathsf{M',M'}} \circ \mathsf{g}$$

となる．ここで，isum″ = $[\![$ p″, id, q″ $]\!]_{\mathsf{M',M'}}$ が isum と等しくなることは容易に確かめられる．結局，以下が導出された．

(c +) ∗ (isum xs d) = isum xs (c + d)

5. 両変換の対応

以上の変換過程を注意深くながめると，両者には密接な対応関係があることに気がつく．表 1 に，*isum* (あるいは isum) の二番目の定義節 (Cons の場合) に注目して両者の関係をまとめた．高階 catamorphism による変換の各段階に，媒介型による変換が対応していることがわかるであろう．これと同様の考察によって，一般的な場合においても「高階 catamorphism で変換可能ならば媒介型を用いた変換も可能であり，両者は同じ結果を導く」ことを示すことができる．

参考文献

[1] Hu, Z., Iwasaki, H. and Takeichi, M.: Catamorphism Based Transformation of Functional Programs, 情報処理学会 プログラミング－言語・基礎・実践－研究会研究報告 16–7, pp.49–56 (1994).

[2] Hu, Z., Iwasaki, H. and Takeichi, M.: Deriving Structural Hylomorphisms from Recursive Definitions, *Proc. 1st ACM SIGPLAN International Conference of Functional Programming*, (to appear) (1996)

[3] Meijer, E., Fokkinga, M. and Paterson, R.: Functional Programming With Bananas, Lenses, Envelopes and Barbed Wire, *Proc. FP CA '91* (*LNCS 523*), Springer–Verlag, pp.124–144 (1991).

[4] Takano, A. and Meijer, E.: Shortcut Deforestation in Calculational Form, *Proc. FPCA '95*, ACM Press, pp.306–313 (1995).

[5] 高野明彦: 構成的アルゴリズム論に基づくプログラム融合変換, 日本ソフトウェア科学会第 12 回大会論文集, pp.265–268 (1995).