蓄積引数を持つ関数プログラムの融合変換

岩崎英哉[†] 胡振江[††]

関数プログラムでは，関数合成間で受け渡される中間的データ構造を生成しないように，関数合成を1つの関数に融合する（融合変換する）ことが，効率改善にとって重要である．構成的アルゴリズム論は，Catamorphism，Hylomorphism などと呼ばれる特定の再帰パターンと，これらに対する強力な変換定理を用いて，上の問題の解決を試みる．本論文は，蓄積引数を持つ関数の融合変換のための2つの手法 —— 高階関数の Catamorphism に基づく方法と Hylomorphism に基づく方法 —— に注目し，前者に基づく融合変換操作は，後者による融合変換に帰着できることを示す．まず簡単な具体例を通して両手法の変換手順を示した後，一般の場合について上の主張を証明する．本論文で示される事実により，構成的アルゴリズム論に基づくプログラム変換システムでは，Hylomorphism を用いた変換だけを考えればよいという，有益な結論が得られる．

Promotional Transformation of Functional Programs with Accumulative Parameter

HIDEYA IWASAKI[†] and ZHENJIANG HU[††]

In order to gain the efficiency of functional programs, it is important to fuse the composition and eliminate intermediate data structures passed through the composition. Constructive Algorithmics is one of the promising approaches to this problem. This paper focuses on functions with accumulative parameters, and discusses two transformation strategies — one based on higher-order catamorphisms and the other based on hylomorphisms. It is shown that programs which can be transformed by the former can be also transformed by the latter. The result of this paper shows that program transformation system based on Constructive Algorithmics has only to take into account the strategy by hylomorphisms.

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．1つ1つの部品（小さな関数）は汎用性に富む効率の良い部品であっても，これらを組み合わせた全体は，そのままでは必ずしも実行効率の良いものになるとは限らない．その大きな原因としては，

- 部品となる関数間で受け渡される多数の中間データ構造がメモリに割り当てられることがあること；
- このようなデータ構造を部品関数がたどる手間が生じること；

の2つをあげることができる．したがって，関数間で受け渡されるが最終結果には出現しない中間的データ構造を生成しないように，関数合成を1つの関数に融合する（これを“融合変換”という）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[1)~6)]の立場からこれらの問題の解決を試みる研究が盛んに行われている．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特定の再帰パターンを持つ関数（以後，これらを Cata，Ana，Hylo と略記する）と，これらの上の様々なプログラム変換定理が中心的な役割を果たす．

部品となる基本関数には，蓄積引数を持つもの[7)]も多数ある．たとえば，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 $isum$ の定義は，

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

で与えられる．ここで d は蓄積引数であり，その初期値としては，一般的には次のように 0 が与えられる．

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

† 東京農工大学工学部電子情報工学科
Department of Computer Science, Tokyo University of Agriculture and Technology
†† 東京大学大学院工学系研究科情報工学専攻
Department of Information Engineering, University of Tokyo

これらの関数を構成的アルゴリズム論で扱うための方法の1つに，高階関数の利用[3]がある．たとえば二引数関数（蓄積引数は第2引数とする）ならば，これを，引数（第1引数）を1つとって一引数関数を返すCataと解釈できる場合がある．関数を値として返す（高階関数）ので，このようなCataを“高階Cata”と呼ぶ．高階Cataは返す値が関数である点を除けば通常のCataと同じなので，Cata上の様々な変換定理を適用することが可能である．

一方，単純なCataでは表現できないような関数を扱うための技法として，中間的なデータ構造を導入する変換[6]も研究されている．二引数関数の場合は，これを2つの引数から別の型（これを“媒介型”と呼ぶ）のデータを作る関数と，その型のデータ上のCataとの合成（4章で述べるが，これは実はHyloであることが示されている）で表現する．その結果，Hylo上のプログラム変換定理を適用することができる．

Hyloを用いた変換が可能な関数の範囲は，高階Cataで表現し変換可能な関数の範囲よりも広い．たとえば，2つの昇順リストを併合した昇順リストを返す関数 $merge$ は，$x > y$ のときを考えると，次のような形の定義節を持つ．これはHyloで表現可能であるが，高階Cataを用いては表現できない（3章参照）．

$$merge\,(x : xs)\,(y : ys)$$
$$= \ldots\ y : merge\,(x : xs)\,ys\ \ldots$$

本論文は逆の関係に注目し，“蓄積引数を持つ関数の高階Cataによるプログラム融合変換操作は，媒介型を用いたHylo上の変換操作に帰着することができる”ことを示す．この事実により，実用面における以下のような有益な結論が得られる．

- 構成的アルゴリズム論に基づくプログラム変換システムを構築する際に，高階Cataによる変換の可能性を考慮する必要はなく，Hyloを用いた変換だけを考えればよい．
- 値として返された関数の定義中にふみこんでのプログラム変換操作は，整数，リストなどの通常の値上の変換操作よりも複雑になる．本論文で示す事実より，関数値を直接扱うことが少なくてすむ．

本論文の構成は，2章で，構成的アルゴリズム論の概要について簡単に触れ，Cata，Ana，Hyloなどの概念，融合変換のための基本的な定理を示す．3，4章では，上で定義を示した $isum$ を用い，$isum\,xs\,d$ の結果に“各要素に c を加える関数” $h = ((c+)*)$ [☆1]を適用した式 $(c+) * (isum\,xs\,d)$ の融合変換[☆2]を具体例として考える（c は適当な定数とする）．この例に即して，高階Cataを用いる手法およびHyloを用いる手法の双方の変換手順と変換定理を示し，互いの関係を明らかにする．5章では，一般的な場合について，高階Cataによる手法はHyloを用いる手法の特別な場合に帰着できることを示す．6章はまとめとする．

3章以降の関数，変数については，高階Cataによる変換には $isum$ という字体を，Hyloによる変換には isum という字体を用い，どちらの方法に基づく変換かを明確にした．

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，

$$\mathsf{L}\,a = \mathsf{Nil} \mid \mathsf{Cons}\,(a, \mathsf{L}\,a)$$

と定義される．ここでNilとConsはデータ構成子[☆3]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次のように定義される．

$$f\,(\mathsf{Nil}\,()) = e\,()$$
$$f\,(\mathsf{Cons}\,(x, xs)) = x \oplus (f\ xs)$$

ここでは，定数（Nil・e の値）を「() を引数とする関数」として表現した．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Cata）を $(\!|e \triangledown \oplus|\!)_{\mathsf{L}}$ と書く（添字は以下で説明する“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[1),2)]（Functor）によって特徴付けられる．本論文における関手は，以下の4つの基本操作[☆4]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．さらに，直和における1，2はタグを表す．

(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） $X + Y = \{1 \quad \times X \cup \{2 \quad \times Y$
$$(f + g)\,(1, x) = (1, f\,x)$$
$$(f + g)\,(2, x) = (2, g\,x)$$
$$(f \triangledown g)\,(1, x) = f\,x,\ (f \triangledown g)\,(2, x) = g\,x$$

☆1 $*$ はmapを表す二項演算子である．また，二項演算子 $\oplus$ に対して，一引数関数 $(a\oplus)$ と $(\oplus b)$ を $(a\oplus)\,b = (\oplus b)\,a = a \oplus b$ で定義する．この括弧付けの操作をセクション化と呼ぶ．

☆2 結果は $isum\,xs\,(c+d)$ となることが期待される．

☆3 本論文ではこれらのデータ構成子のかわりに，通常のリストの記法（$[]$，$x : xs$，$[1, 2]$）を用いることもある．

☆4 積と直和は2パラメータのものを示したが，これらの定義は自然に n パラメータに拡張することができる．

要素が a 型のリスト型の場合の関手 L_a は，$\mathbf{1}$ を "() のみを要素とする集合" とすると，$\mathsf{L}_a = !\mathbf{1} + !a \times \mathsf{I}$ となる．これに p を関数として適用すると $\mathsf{L}_a p = id + id \times p$ となる．本論文では，型名（型構成子）とその型に対応する関手☆を，同じ記号で表すこととする．

本節の冒頭で定義したリスト上の Cata の一般形 f の定義式を書きかえる☆☆と，

$$\begin{aligned}(f \circ \mathsf{Nil})\,() &= e\,()\\ &= (e \circ id)\,()\\ (f \circ \mathsf{Cons})\,(x, xs) &= x \oplus (f\ xs)\\ &= (\oplus \circ (id \times f))\,(x, xs)\end{aligned}$$

となる．引数が () の場合と (x, xs) の場合とを区別するため，$(1, ())$，$(2, (x, xs))$ のようにタグをつけ，上式を 1 つにまとめると次のようになる．

$$\begin{aligned}f \circ (\mathsf{Nil} \triangledown \mathsf{Cons}) &= (e \triangledown \oplus) \circ (id + id \times f)\\ &= (e \triangledown \oplus) \circ \mathsf{L}\,f\end{aligned}$$

$\mathsf{Nil} \triangledown \mathsf{Cons}$ を in_{L} と表記することにすれば，$f = (\!|e \triangledown \oplus|\!)_{\mathsf{L}}$ であるから上式は，

$$(\!|e \triangledown \oplus|\!)_{\mathsf{L}} \circ in_{\mathsf{L}} = (e \triangledown \oplus) \circ \mathsf{L}\,(\!|e \triangledown \oplus|\!)_{\mathsf{L}}$$

となる．これがリスト上の Cata を特徴付ける式である．上の議論を見ると，関手に関数を適用した $\mathsf{L}\,f$ には，"型 L に属すデータの再帰部分に f の適用を分配する" という作用があることが分かるであろう．

一般的には，型 F 上の Cata である $(\!|\phi|\!)_{\mathsf{F}}$ 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 で定義される型のデータ構成子である．

2.2 融合定理

Cata におけるプログラム融合変換の基本となるのが，融合定理[1),2)]である．

定理 1 （融合定理）

$$h \circ \phi = \phi' \circ \mathsf{F}\,h \qquad \text{(A)}$$
$$\Longrightarrow h \circ (\!|\phi|\!)_{\mathsf{F}} = (\!|\phi'|\!)_{\mathsf{F}} \qquad \square$$

リストの場合（$\phi = e \triangledown \oplus$），条件 (A) は次を満たす e'，$\otimes$ の存在（$\phi' = e' \triangledown \otimes$）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 Cata との合成が別の Cata 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，関数合成を 1 つに融合することができ，プログラムの実行効率の改善が期待できる．

2.3 Anamorphism と Hylomorphism

Cata 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 のデータを自然な形で再帰的に生成する関数を考えることもできる．このような関数を Anamorphism と呼ぶ．たとえば，与えられた引数 n に対して $[\mathtt{n}, \mathtt{n}-1, \ldots, 2, 1]$ を返す関数

```
downto1 n =
    case n of 0 → Nil
              m + 1 → Cons (n, downto1 m)
```

は，リストを生成する Ana である．ここで関数 ψ を，

$$\begin{aligned}\psi\,\mathtt{n} = \mathsf{case}\ \mathtt{n}\ \mathsf{of}\ &0 \to (1, ())\\ &\mathtt{m}+1 \to (2, (\mathtt{n}, \mathtt{m}))\end{aligned}$$

で定義すると，downto1 の定義式の右辺は，

$$((\mathsf{Nil} \triangledown \mathsf{Cons}) \circ (id + id \times \mathtt{downto1}) \circ \psi)\,\mathtt{n}$$

と変形することができ，

$$\mathtt{downto1} = in_{\mathsf{L}} \circ \mathsf{L}\,\mathtt{downto1} \circ \psi$$

となる．downto1 は ψ によって一意に定められるので，これを $[\!(\psi)\!]_{\mathsf{L}}$ という記法で表す．

in_{L} の逆関数を out_{L} とすると，上式は，

$$out_{\mathsf{L}} \circ [\!(\psi)\!]_{\mathsf{L}} = \mathsf{L}\,[\!(\psi)\!]_{\mathsf{L}} \circ \psi$$

と表現される．ここで out_{L} はリストの分解子である．

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

一般的には $[\!(\psi)\!]_{\mathsf{F}}$（型 F のデータを生成する Ana）は，$out_{\mathsf{F}}$ をデータの分解子（in_{F} の逆関数）として，

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

という式によって特徴付けられる．これは 2.1 節の最後に述べた Cata の特徴付けの式に対応している．

さらに，Cata と Ana との合成 $(\!|\phi|\!)_{\mathsf{F}} \circ [\!(\psi)\!]_{\mathsf{F}}$ を Hylomorphism[2)] と呼び，$[\![\phi, \psi]\!]_{\mathsf{F}}$ という記法を用いることがある．Hylo には，η を F から G への自然変換（η が F から G への自然変換とは，任意の f について $\eta \circ \mathsf{F}\,f = \mathsf{G}\,f \circ \eta$ が成立することをいう）とするとき，Hylo Shift と呼ばれる次の性質がある．

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

この性質は，"二つ組" の Hylo 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．そこで本論文では，自然変換をくくり出した "三つ組" の Hylo の表現[4)] 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，$[\![\phi, \eta, \psi]\!]_{\mathsf{G},\mathsf{F}}$ と表される．二つ組表現における Hylo Shift は次のとおり．

$$\begin{aligned}[\![\phi \circ \eta, \mathtt{id}, \psi]\!]_{\mathsf{F},\mathsf{F}} &= [\![\phi, \eta, \psi]\!]_{\mathsf{G},\mathsf{F}}\\ &= [\![\phi, \mathtt{id}, \eta \circ \psi]\!]_{\mathsf{G},\mathsf{G}}\end{aligned}$$

Hylo はその一部に Cata と Ana を含むので，Cata における融合定理に相当するものと，その双対定理（Ana における融合定理）が成立する．

定理 2 （Hylo 融合定理）

$$h \circ \phi = \phi' \circ \mathsf{G}\,h$$

☆ 式が繁雑になるのを避けるため，以後本論文では関手の添字（パラメータ）を省略する．

☆☆ $\circ$ は関数合成であり，関数適用よりも結合の優先度が低い．

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

$$\psi \circ g = \mathsf{F}\, g \circ \psi'$$

$$\Longrightarrow [\![\phi,\eta,\psi]\!]_{\mathsf{G},\mathsf{F}} \circ g = [\![\phi,\eta,\psi']\!]_{\mathsf{G},\mathsf{F}} \qquad \square$$

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，二引数のある種の関数に関しては“引数を1つもらい，関数を返す高階関数”としてとらえれば，通常の Cata の枠組で説明することができる場合もある．このような Cata を“高階 Cata”と呼ぶ．たとえば *isum* はリスト上の高階 Cata である（3.2.1 項参照）．

一方，*merge* は高階 Cata では表現できない．なぜなら，Cata になるためには引数を必ず消費しなければならないが，*merge* では引数の $x : xs$ が消費されずに，そのままの形で再帰呼び出しの引数として渡されているためである．

3.1 高階融合定理

高階 Cata に関する融合定理は，以下のようになる．$h\,([\![\phi]\!]_{\mathsf{F}}\, x\, d)$ という式で蓄積引数の d 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 $(h\circ)$ に対応することが分かる．よって次の定理が得られる．

定理 3 （高階融合定理 1）

$$(h\circ)\circ\phi = \phi'\circ \mathsf{F}\,(h\circ) \qquad \text{(B)}$$

$$\Longrightarrow (h\circ)\circ[\![\phi]\!]_{\mathsf{F}} = [\![\phi']\!]_{\mathsf{F}} \quad \text{すなわち}$$

$$h\,([\![\phi]\!]_{\mathsf{F}}\, x\, d) = [\![\phi']\!]_{\mathsf{F}}\, x\, d \qquad \square$$

リストの場合，$\phi = e \triangledown \oplus$，$\phi' = e' \triangledown \otimes$ として，

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

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，この定理では蓄積引数に対する“操作”を加えることができない（融合変換前後で蓄積引数は d のまま変化しない）ためである．したがって，融合結果の Cata から関数を抽出して蓄積引数に適用させるような定理が必要である．

定理 4 （高階融合定理 2）

$$(\circ g)\circ\phi = \phi'\circ \mathsf{F}\,(\circ g) \qquad \text{(B')}$$

$$\Longrightarrow (\circ g)\circ[\![\phi]\!]_{\mathsf{F}} = [\![\phi']\!]_{\mathsf{F}} \quad \text{すなわち}$$

$$[\![\phi]\!]_{\mathsf{F}}\, x\, (g\, d) = [\![\phi']\!]_{\mathsf{F}}\, x\, d \qquad \square$$

この定理の帰結部を右辺から左辺に向かって読むと，高階 Cata から蓄積引数部へ関数 (g) を抽出するための規則として解釈することができる．

リストの場合（$\phi = e'' \triangledown \ominus$, $\phi' = e' \triangledown \otimes$）は，次式を満たすことが条件 (B') となる．

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

まとめると，ある関数と高階 Cata の合成は，高階融合定理 1 による融合変換で別の高階 Cata に変換し，さらに高階融合定理 2 で蓄積引数部に関数を抽出する．この結果，効率の良いプログラムに変換される．

3.2 *isum* 上の融合変換

高階 Cata を用いた $h\,(isum\, xs\, d)$ の変換（$h = ((c+)*)$）は，以下のような手順で進む．

(1) *isum* を高階 Cata 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 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 との融合

高階融合定理 1 より，$(h\circ)\circ p = p'\circ \mathsf{L}\,(h\circ)$ となる p' を探す必要がある．

$p' = p'_1 \triangledown p'_2$ とおくと，この式は，

$$(h\circ)\circ(p_1 \triangledown p_2) = (p'_1 \triangledown p'_2)\circ(id + id \times (h\circ))$$

すなわち，次のように変形できる．

$$((h\circ)\circ p_1) \triangledown ((h\circ)\circ p_2) = p'_1 \triangledown (p'_2 \circ (id \times (h\circ)))$$

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 を見つける．

$$\text{左辺} = h\,(p_1\,()\, d)$$

$$= [c + d]$$

なので，次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 を見つける．

$$\text{左辺} = h\,(p_2\,(x, r)\, d)$$

$$= (c+) * (d : r\,(x + d))$$

$$= (c + d) : ((c+) * (r\,(x + d)))$$

$$= (c + d) : ((h \circ r)\,(x + d))$$

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 $h\,([\![p]\!]_{\mathsf{L}}\, xs\, d) = [\![p']\!]_{\mathsf{L}}\, xs\, d$ と融合変換された．

3.2.3 g の抽出

高階融合定理 2 に基づき，$(\circ g)\circ p'' = p'\circ \mathsf{L}\,(\circ g)$ となる p'' と g を探す．$p'' = p''_1 \triangledown p''_2$ とおくと，

$((\circ g)\circ p_1'')\triangledown((\circ g)\circ p_2'') = p_1'\triangledown(p_2'\circ(id\times(\circ g)))$

と変形される．

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'' を見つける．

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

よって，$g\,x = c+x$ と定義すると次式となる．

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

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'' を見つける．

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

よって，次の式が導かれる．

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

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 $(\![p'']\!)_{\mathsf{L}}$ は $(\![p]\!)_{\mathsf{L}}$ すなわち $isum$ と等しい．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. Hylomorphism による変換

4.1 媒介型の導入

Hylo による変換では，複数の引数をカリー化せず組として一体化し，単一引数とする方が都合がよい．そこで次のような `isum` の定義を出発点とする．

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

ところが（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 Cata や Hylo 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”[6]する．

具体的には，注目する関数 $\mathtt{f} :: \mathsf{F}\times\mathsf{G}\to\mathsf{T}$ に対して，適当な型 M（“媒介型”と呼ぶ）を考え，f を $\mathtt{f}' :: \mathsf{M}\to\mathsf{T}$ と $\mathtt{t} :: \mathsf{F}\times\mathsf{G}\to\mathsf{M}$ との合成 $\mathtt{f} = \mathtt{f}'\circ\mathtt{t}$ として表現する．ここで，t は f の引数を M 型に変換するような型変換関数である．$\mathtt{f}'$ が $(\![\phi]\!)_{\mathsf{M}}$（M 上の Cata）として表現できれば，融合定理を利用して f を含むプログラムを構成的に変換することが可能となる．以上の変換には，次の性質があることが示されている．

- 型変換関数 t は 実は M 型のデータを生成する Ana として表現することができる．
- 中間に現れる媒介型 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，$\mathtt{t} = [\![\psi]\!]_{\mathsf{M}}$ とおくと，f は

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

と表現される．

対象とする関数が Hylo で表現されれば，あとは Hylo 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 Hylo の内部を自由に移動できる性質（Hylo Shift）を利用し，Hylo 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，Hylo 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重要である．このように Hylo の内側で自然変換を必要に応じて移動することを“Hylo の再構成”と呼ぶ．

与えられた関数の定義から媒介型と Hylo を導出するアルゴリズム，Hylo の再構成アルゴリズムの詳細は，文献 6) を参照されたい．

4.2 isum 上の融合変換

h∘isum の変換の具体的手順は，以下のようになる．ここで，$\mathtt{h} = ((\mathtt{c}+)*)$ である．

(1) 媒介型 M を導入して isum を Hylo で表現し，上式を $\mathtt{h}\circ[\![\mathtt{p},\mathtt{id},\mathtt{q}]\!]_{\mathsf{M},\mathsf{M}}$ とする．

(2) 上式で h を融合して $[\![\mathtt{p}',\mathtt{id},\mathtt{q}]\!]_{\mathsf{M},\mathsf{M}}$ とする．

(3) $\mathtt{p}' = \mathtt{p}''\circ\mathtt{n}$ となるような自然変換 $\mathtt{n} :: \mathsf{M}\to\mathsf{M}'$ を $\mathtt{p}'$ から抽出する．さらに n を Hylo の中で移動すると，$[\![\mathtt{p}'',\mathtt{id},\mathtt{n}\circ\mathtt{q}]\!]_{\mathsf{M}',\mathsf{M}'}$ となる．

(4) Hylo 融合定理の二番目を用いて，関数 g を右側から抽出して，$[\![\mathtt{p}'',\mathtt{id},\mathtt{q}'']\!]_{\mathsf{M}',\mathsf{M}'}\circ\mathtt{g}$ とする．

以上より，次式が導かれる．

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

4.2.1 isum の Hylomorphism 表現

$\mathtt{isum} = \mathtt{isum}'\circ\mathtt{t}$（ただし $\mathtt{t} :: [\mathsf{Int}]\times\mathsf{Int}\to\mathsf{M}$ は Ana となる型変換関数，$\mathtt{isum}' :: \mathsf{M}\to\mathsf{T}$ は Cata）と変形する．媒介型 M は，以下のように定められる．

- 注目する関数の定義節それぞれに対応する（定義節と同数の）データ構成子を持つ．
- 各データ構成子の再帰的引数の数は対応する定義節の右辺中の再帰呼び出しの数と同じとし，非再帰的引数は，定義節の右辺から再帰呼び出しを除

いた部分に現れる変数と対応するようにする．

isum の場合は，2つの定義節に対応した2つのデータ構成子（D_1, D_2 とする）を用意する．isum ([], d) の右辺には再帰呼び出しがなく d（Int 型）のみが用いられているので，D_1 の引数は Int とする．一方，isum (x : xs, d) の右辺には再帰呼び出しが1つあり，その他の部分では d だけが用いられているので，D_2 の引数は (Int, M) とする．したがって M は，

$$M = D_1\ \mathsf{Int} \mid D_2\ (\mathsf{Int}, M)$$

と定義され，この型を定める関手 M は $M\phi = \mathrm{id} + \mathrm{id} \times \phi$ となる．t の定義は，次のように与えられる．

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

実はこれは M 上の Ana である．実際，

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

とすると，$t = [\![q]\!]_M$ と表現できる．一方，isum′ :: M → [Int] は次のような Cata 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，$isum = [\![p, \mathrm{id}, q]\!]_{M,M}$ という Hylo で表現できたことになる．

4.2.2 h との融合

Hylo 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．$p' = p'_1 \triangledown p'_2$ とすると，この式は，

$$(h \circ p_1) \triangledown (h \circ p_2) = p'_1 \triangledown (p'_2 \circ (\mathrm{id} \times h))$$

となる．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 を見つける．

$$\text{左辺} = (c +) * [d]$$
$$= [c + d]$$

なので，$p'_1\ d = [c + d]$ と定めることができる．

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

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

となる p'_2 を探す．

$$\text{左辺} = (c +) * (d : r)$$
$$= (c + d) : ((c +) * r)$$

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

以上の議論から次のように融合された．

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

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．これらは以下のようにして定められる．

- p' の定義中，再帰に対応する変数（再帰変数，p'_2 の場合は r）が現れないような“非再帰部”の計算をなるべく大きく取り出して n の中へ移動する．
- p'' は n に移した非再帰部の計算結果を受け，再帰変数に関連した処理と合わせて結果とする．

$p'_1\ d = [c + d]$ は再帰変数を持たず，右辺全体が非再帰部であるから，これをそのまま自然変換部に移す．そこで，次のように p''_1, n_1 を定める．

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

$p'_2\ (d, r)$ の右辺では，r が再帰変数なので，(c + d) が非再帰部となる．そこで，次のように定める．

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

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 は M から次のように定義される型 M′ への自然変換となる．ただし，D'_1, D'_2 は M′ のデータ構成子である．

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

この型を定める関手 M′ は $M'\phi = \mathrm{id} + \mathrm{id} \times \phi$ となる．最後に n を Hylo 内を右に移動して $q' = n \circ q$ とおくことによって，次式が導かれる．

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

4.2.4 g の抽出

$q' = n \circ q$ に従って q' を書き下すと次のとおり．

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

Hylo 融合定理より，$M'\,g \circ q' = q'' \circ g$ すなわち

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

となる g, q'' を見つけたい．

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

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

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

となり，これが $(q'' \circ g)\ (xs, d) = q''\ (xs, c + d)$ と等しくならなければならない．よって q'' は

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

と定義すればよい．以上すべてをまとめると，

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

となる．ここで，$[\![\mathtt{p}'', \mathtt{id}, \mathtt{q}'']\!]_{\mathsf{M}',\mathsf{M}'}$ が isum と等しくなることは容易に確かめられる．結局，次式が導出され，高階 Cata の場合と実質的に同じ結果が得られた．

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

5. 両変換の対応

本章では一般的な場合について，次の性質を示す．

定理 5 蓄積引数を持つ関数の融合変換が高階 Cata を用いて可能ならば，Hylo を用いた変換も可能であり，両者は同じ結果を導く． □

前章までと同様に，用いる字体によってどちらの変換の話なのかを区別し，両変換の手順を並べて説明する．また v ($\mathtt{v}$) は，高階 Cata による変換の場合には v，Hylo による変換の場合には $\mathtt{v}$ を意味するものとする．

蓄積引数を持つ関数 $f :: \mathsf{F} \to \mathsf{G} \to \mathsf{T}$ （Hylo の場合には $\mathtt{f} :: \mathsf{F} \times \mathsf{G} \to \mathsf{T}$) を考える．型 F の定義は，

$$\mathsf{F} = \cdots \mid \mathsf{C}_i\,(\mathsf{Z}_1, \cdots, \mathsf{Z}_k, \mathsf{F}, \cdots, \mathsf{F}) \mid \cdots$$

（右辺の F の個数は l 個）で与えられる．

以下では，次の約束に基づいた記法を用いる．

- 添字 i は，型の i 番目の定義（上の例ではデータ構成子 C_i で始まる定義）に関する事項を表す．たとえば，F_i は 関手 F の i 番目の成分 ($\mathsf{F} = \cdots + \mathsf{F}_i + \cdots$)，すなわち，$\mathsf{C}_i$ で始まる定義に対応する関手 $\mathsf{F}_i = !\mathsf{Z}_1 \times \cdots \times !\mathsf{Z}_k \times \mathsf{I} \times \cdots \times \mathsf{I}$ を表す（I は l 個 ）．
- 型あるいは関数の引数の “再帰部” に相当する変数には r （$\mathtt{r}$）を，それ以外（非再帰部）に相当する変数には v ($\mathtt{v}$) を主として用いる．さらに，蓄積引数には d ($\mathtt{d}$) を用いる．
- 添字 j を，“j を文脈上考えうるすべての範囲にわたって動かした場合のその添字を含む式の並び” を表すものとして用いる．たとえば，$\mathsf{C}_i\,(v_j, r_j)$ は $\mathsf{C}_i\,(v_1, \cdots, v_k, r_1, \cdots, r_l)$ を，p_i の定義域が C_i と同じとき，$p_i\,(v_j, h\ r_j)$ は $p_i\,(v_1, \cdots, v_k, h\,r_1, \cdots, h\,r_l)$ を表す．これらで，v_j が k 個の並びに，r_j と $h\,r_j$ が l 個の並びになるのは，型 F の定義に対応している．さらに，ある式の中の複数の j を同時に動かすことを明示する場合には，その式に下線を引いて表す．たとえば，$\underline{(x_j, y_j)}$ は $(x_1, y_1), (x_2, y_2), \cdots$ を表す．

上の記法に従えば，f ($\mathtt{f}$) の定義は，一般的には以下のように表現することができよう．ここで，α_j は再帰呼び出しにおける蓄積引数を更新する関数である．ただし，v_j ($\mathtt{v}_j$) は k 個の並び，r_j ($\mathtt{r}_j$) に関する部分は l 個の並びである．

$$f\,(\mathsf{C}_i\,(v_j, r_j)) = \lambda d\,.\,\mathcal{E}_i\,(v_j,\ d,\ \underline{f\ r_j\ (\alpha_j\ d)})$$
$$\mathtt{f}\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j),\ \mathtt{d}) = \mathcal{E}_i\,(\mathtt{v}_j,\ \mathtt{d},\ \underline{\mathtt{f}\,(\mathtt{r}_j,\ \alpha_j\ \mathtt{d})})$$

問題は，$h\,(f\,x\,d)$ と $\mathtt{h}\,(\mathtt{f}\,(\mathtt{x}, \mathtt{d}))$ が与えられたときの，両者の融合変換操作の関係を明らかにし，前者で融合変換が可能ならば後者においても可能であることを示すことである．**表 1** には，以降で説明する両変換の関係の要点をまとめておく．

5.1 Catamorphism と Hylomorphism 表現

はじめに f の Cata 表現を考える．上の f の定義より，$f = (\!|p|\!)_{\mathsf{F}}$，$p = \cdots \triangledown p_i \triangledown \cdots$ とすると，

$$p_i\,(v_j, r_j) = \lambda d\,.\,\mathcal{E}_i\,(v_j,\ d,\ \underline{r_j\ (\alpha_j\ d)})$$

となる．

Hylo を用いた変換では，型変換関数 $\mathtt{t} :: \mathsf{F} \times \mathsf{G} \to \mathsf{M}$，$\mathtt{f}' :: \mathsf{M} \to \mathsf{T}$ によって，$\mathtt{f} = \mathtt{f}' \circ \mathtt{t}$ と表現する．ただし，$\mathtt{f}'$ は Cata，$\mathtt{t}$ は Ana である（$\mathtt{f}' = (\!|\mathtt{p}|\!)_{\mathsf{M}}$，$\mathtt{t} = [\![\mathtt{q}]\!]_{\mathsf{M}}$）．ここでは一般性を失わず説明を簡単にするため，$\mathcal{E}_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j)$ から再帰呼び出しを除いた部分の自由変数には，$\mathtt{v}_j$ と $\mathtt{d}$ がすべて含まれるものと仮定する．すると，$\mathtt{v}_j$ の型が Z_j，$\mathtt{d}$ の型が G，再帰呼び出しが l 個あることを考慮すると，媒介型 M の定義は，D_i をデータ構成子として次のようになるであろう．

$$\mathsf{M} = \cdots \mid \mathsf{D}_i\,(\mathsf{Z}_1, \cdots, \mathsf{Z}_k, \mathsf{G}, \mathsf{M}, \cdots, \mathsf{M}) \mid \cdots$$

これに対応する関手 M は，$\mathsf{M}_i = !\mathsf{Z}_1 \times \cdots \times !\mathsf{Z}_k \times !\mathsf{G} \times \mathsf{I} \times \cdots \times \mathsf{I}$ である （I は l 個）．

さて，$\mathtt{t}$ と $\mathtt{f}'$ の定義は

$$\mathtt{t}\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j),\ \mathtt{d}) = \mathsf{D}_i\,(\mathtt{v}_j,\ \mathtt{d},\ \underline{\mathtt{t}\,(\mathtt{r}_j,\ \alpha_j\ \mathtt{d})})$$
$$\mathtt{f}'\,(\mathsf{D}_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j)) = \mathcal{E}_i\,(\mathtt{v}_j,\ \mathtt{d},\ \mathtt{f}'\ \mathtt{r}_j)$$

で与えられるので，$\mathtt{p} = \cdots \triangledown \mathtt{p}_i \triangledown \cdots$ とすると，

$$\mathtt{q}\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j),\ \mathtt{d}) = (i,\ (\mathtt{v}_j,\ \mathtt{d},\ \underline{(\mathtt{r}_j,\ \alpha_j\ \mathtt{d})}))$$
$$\mathtt{p}_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j) = \mathcal{E}_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j)$$

である．よって，$\mathtt{f} = [\![\mathtt{p}, \mathtt{id}, \mathtt{q}]\!]_{\mathsf{M},\mathsf{M}}$ という Hylo で表現された．

5.2 h (h) の融合

まず高階 Cata の場合を考える．$h\,((\!|p|\!)_{\mathsf{F}}\ x\ d) = (\!|p'|\!)_{\mathsf{F}}\ x\ d$ として，$p' = \cdots \triangledown p'_i \triangledown \cdots$ とおくと，高階融合定理 1 に基づき $(h \circ) \circ p_i = p'_i \circ \mathsf{F}_i(h \circ)$ となる p'_i を探す必要がある．両辺に (v_j, r_j) と d を引数として与えると，

$$\text{左辺} = h\,(p_i\,(v_j, r_j)\ d)$$
$$= h\,(\mathcal{E}_i\,(v_j,\ d,\ \underline{r_j\ (\alpha_j\ d)})) \qquad (*)$$
$$\text{右辺} = p'_i\,(v_j, h \circ r_j)\ d$$

である．ここで p'_i は，

$$p'_i\,(v_j, r_j) = \lambda d\,.\,\mathcal{E}'_i\,(v_j,\ d,\ \underline{r_j\ (\alpha_j\ d)})$$

という形になるはずである．なぜなら，(*) 式において h を $\mathcal{E}_i$ の中に融合する際，r_j はもともと p_i の

表1 高階 Cata による変換と Hylo による変換との対応
Table 1 Relationship between two transformations.

	高階 Cata による変換	Hylo による変換	
f の定義	$f\,(\mathsf{C}_i\,(v_j, r_j)) =$ $\lambda d\,.\,\mathcal{E}_i\,(v_j,\ d,\ \underline{f\ r_j\ (\alpha_j\ d)})$	$\mathtt{f}\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j),\ \mathtt{d}) =$ $\mathcal{E}_i\,(\mathtt{v}_j,\ \mathtt{d},\ \underline{\mathtt{f}\,(\mathtt{r}_j,\ \alpha_j\ \mathtt{d})})$	f の定義
Cata	$f\ x\ d = (\!\lvert p \rvert\!)_{\mathsf{F}}\ x\ d$ $p_i\,(v_j, r_j) =$ $\lambda d\,.\,\mathcal{E}_i\,(v_j,\ d,\ \underline{r_j\ (\alpha_j\ d)})$	$\mathtt{f}\,(\mathtt{x}, \mathtt{d}) = [\![\mathtt{p}, \mathtt{id}, \mathtt{q}]\!]_{\mathsf{M},\mathsf{M}}\,(\mathtt{x}, \mathtt{d})$ $\mathtt{p}_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j) =$ $\mathcal{E}_i\,(\mathtt{v}_j,\ \mathtt{d},\ \mathtt{r}_j)$ $\mathtt{q}\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d}) =$ $(i,\ (\mathtt{v}_j,\ \mathtt{d},\ \underline{(\mathtt{r}_j, \alpha_j\ \mathtt{d})}))$	Hylo
h の融合	$h\,((\!\lvert p \rvert\!)_{\mathsf{F}}\ x\ d) = (\!\lvert p' \rvert\!)_{\mathsf{F}}\ x\ d$ $p'_i\,(v_j, r_j) =$ $\lambda d\,.\,\mathcal{E}'_i\,(v_j,\ d,\ \underline{r_j\ (\alpha_j\ d)})$	$\mathtt{h}\,([\![\mathtt{p}, \mathtt{id}, \mathtt{q}]\!]_{\mathsf{M},\mathsf{M}}\,(\mathtt{x}, \mathtt{d})) =$ $[\![\mathtt{p}', \mathtt{id}, \mathtt{q}]\!]_{\mathsf{M},\mathsf{M}}\,(\mathtt{x}, \mathtt{d})$ $\mathtt{p}'_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j) =$ $\mathcal{E}'_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j)$	h の融合
		$[\![\mathtt{p}', \mathtt{id}, \mathtt{q}]\!]_{\mathsf{M},\mathsf{M}} = [\![\mathtt{p}'', \mathtt{id}, \mathtt{q}']\!]_{\mathsf{M}',\mathsf{M}}$ $\mathtt{p}''_i\,(\mathtt{u}_j, \mathtt{r}_j) = \mathcal{E}''_i\,(\mathtt{u}_j, \mathtt{r}_j)$ $\mathtt{q}'\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d}) =$ $(i,\ (t_j\ (\mathtt{d}),\ \underline{(\mathtt{r}_j,\ \alpha_j\ \mathtt{d})}))$	再構成
g の抽出	$(\!\lvert p' \rvert\!)_{\mathsf{F}}\ x\ d = (\!\lvert p'' \rvert\!)_{\mathsf{F}}\ x\ (g\ d)$ $p''_i\,(v_j, r_j) =$ $\lambda d\,.\,\mathcal{E}''_i\,(t'_j\,(d),\ \underline{r_j\ (\beta_j\ d)})$	$[\![\mathtt{p}'', \mathtt{id}, \mathtt{q}']\!]_{\mathsf{M}',\mathsf{M}'}\,(\mathtt{x}, \mathtt{d}) =$ $[\![\mathtt{p}'', \mathtt{id}, \mathtt{q}'']\!]_{\mathsf{M}',\mathsf{M}'}\,(\mathtt{x},\ g\ \mathtt{d})$ $\mathtt{p}''_i\,(\mathtt{u}_j, \mathtt{r}_j) =$ $\mathcal{E}''_i\,(\mathtt{u}_j, \mathtt{r}_j)$ $\mathtt{q}''_i\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d}) =$ $(i,\ (t'_j\ (\mathtt{d}),\ \underline{(\mathtt{r}_j,\ \beta_j\ \mathtt{d})}))$ $\mathtt{g}\,(\mathtt{x}, \mathtt{d}) = (\mathtt{x},\ g\ \mathtt{d})$	g の抽出
f'' の定義	$f''\,(\mathsf{C}_i\,(v_j, r_j)) =$ $\lambda d\,.\,\mathcal{E}''_i\,(t'_j\,(d),\ \underline{f''\ r_j\ (\beta_j\ d)})$	$\mathtt{f}''\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d}) =$ $\mathcal{E}''_i\,(t'_j\,(\mathtt{d}),\ \underline{\mathtt{f}''\,(\mathtt{r}_j,\ \beta_j\ \mathtt{d})})$	f'' の定義

引数なので，(*) の変形をさらに進めても $\underline{r_j\ (\alpha_j\ d)}$ の r_j と $(\alpha_j\ d)$ とを分離することはできない．よって (*) 式を変形して $h \circ r_j$ という形を出現させるとすると，それは $\underline{(h \circ r_j)\ (\alpha_j\ d)}$ という形しかありえない．したがって $\mathcal{E}'_i$ を適当な関数として，$(*) = \mathcal{E}'_i\,(v_j,\ d,\ \underline{(h \circ r_j)\ (\alpha_j\ d)})$ と変形されるはずだからである．

Hylo の場合には，$\mathtt{h} \circ \mathtt{p}_i = \mathtt{p}'_i \circ \mathsf{M}_i\,\mathtt{h}$ となる $\mathtt{p}'_i$ を発見すればよい．この式の両辺に $(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j)$ を引数として与えると，次のようになる．

$$\begin{aligned} \text{左辺} &= \mathtt{h}\,(\mathtt{p}_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j)) \\ &= \mathtt{h}\,(\mathcal{E}_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j)) \qquad (**) \\ \text{右辺} &= \mathtt{p}'_i\,(\mathtt{v}_j,\ \mathtt{d},\ \mathtt{h}\ \mathtt{r}_j) \end{aligned}$$

ここで高階 Cata のときの議論より，$\mathtt{h}(\mathcal{E}_i(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j)) = \mathcal{E}'_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j)$ となり，$\mathtt{p}' = \cdots \triangledown \mathtt{p}'_i \triangledown \cdots$ と表すことにすると，$\mathtt{h} \circ [\![\mathtt{p}, \mathtt{id}, \mathtt{q}]\!]_{\mathsf{M},\mathsf{M}} = [\![\mathtt{p}', \mathtt{id}, \mathtt{q}]\!]_{\mathsf{M},\mathsf{M}}$ と融合することができるはずである．

以上より，高階 Cata において h が融合可能ならば，Hylo においても h が融合可能であることが示された．

5.3 g (g) の抽出

高階 Cata の場合には，高階融合定理 2 に基づき，$(\circ g) \circ p''_i = p'_i \circ \mathsf{F}_i(\circ g)$ となる p''_i と g を探す．両辺に (v_j, r_j) と d を引数として与えると，

$$\begin{aligned} \text{右辺} &= p'_i\,(v_j, r_j \circ g)\,d \\ &= \mathcal{E}'_i\,(v_j,\ d,\ \underline{(r_j \circ g)\ (\alpha_j\ d)}) \qquad (\dagger) \\ \text{左辺} &= p''_i\,(v_j, r_j)\,(g\,d) \end{aligned}$$

となる．左辺の形を見ると，(†) 式での d の出現はすべて $g\,d$ の形になるように変換されなければならない．

(†) 式に含まれる d には，次の 2 種類が考えられる．

- $\underline{(r_j \circ g)\ (\alpha_j\ d)}$ として出現するもの；
- $\underline{(r_j \circ g)\ (\alpha_j\ d)}$ に含まれないもの；

前者については，$\underline{(r_j \circ g)\ (\alpha_j\ d)}$ 中の d を，はじめに与えた引数である r_j の外側に出すことができないことを考えると，$\underline{g\ (\alpha_j\ d) = \beta_j\ (g\,d)}$ なる β_j が存在して，$\underline{(r_j \circ g)\ (\alpha_j\ d) = (r_j \circ \beta_j)\ (g\,d)}$ と変形される必要がある．

後者に該当する d の出現それぞれについては，$\mathcal{E}'_i$ 中の d の出現を含むすべての非再帰部が，適当な変換により，$g\,d$ の形でしか d が現れないようにならなければいけない．$\mathcal{E}'_i$ 中の非再帰部（全部で m 個とする）を $t_c\,(v_j, d)$ $(c = 1, \cdots, m)$ とするとき，これは $\underline{t_c\,(v_j, d) = t'_c\,(v_j,\ g\,d)}$ と変換されなければならないことを意味する．以後，記号が繁雑になるのを避けるため，$t_c\,(v_j, d)$ と $t'_c\,(v_j, d)$ の引数はそれぞれ $t_c\,(d)$ と $t'_c\,(d)$ のように蓄積引数のみを記す．

(†) 式は，非再帰部である $t_j\,(d)$ と $\underline{(r_j \circ g)\,(\alpha_j\,d)}$ を引数とするような関数 $\mathcal{E}_i''$ を用いて，

$$(\dagger) = \mathcal{E}_i''\,(t_j\,(d), \underline{(r_j \circ g)\,(\alpha_j\,d)})$$

と表現しなおすことが可能である．

以上の議論を考慮すると (†) 式は

$$(\dagger) = \mathcal{E}_i''\,(t_j'\,(g\,d), \underline{(r_j \circ \beta_j)\,(g\,d)})$$

と書ける．これと左辺とを比較することによって次が得られ，g の抽出が完了する．

$$p_i''\,(v_j, r_j) = \lambda d\,.\,\mathcal{E}_i''\,(t_j'\,(d), \underline{r_j\,(\beta_j\,d)})$$

$p'' = \cdots \triangledown p_i'' \triangledown \cdots$ とおいて，次の関係が判明した．

$$(\![p']\!)_{\mathsf{F}}\,xs\,d = (\![p'']\!)_{\mathsf{F}}\,xs\,(g\,d)$$

次に，Hylo による変換操作を考える．まず，前段階で得られた $[\![\mathtt{p}', \mathtt{id}, \mathtt{q}]\!]_{\mathsf{M},\mathsf{M}}$ を再構成しなければならない．つまり，$\mathtt{p}' = \mathtt{p}'' \circ \mathtt{n}$ となる自然変換 $\mathtt{n} :: \mathsf{M} \to \mathsf{M}'$ を取り出し，これを Hylo 内を右に移動させて，$[\![\mathtt{p}'', \mathtt{id}, \mathtt{n} \circ \mathtt{q}]\!]_{\mathsf{M}',\mathsf{M}}$ と変形する．

まず，$\mathtt{p}_i'\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j) = \mathcal{E}_i'\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j)$ の右辺の非再帰部を取り出す．すでに高階 Cata による変換で仮定したとおり，非再帰部は $t_c\,(d)$ $(c = 1, \cdots, m)$ であった．すると新しい媒介型 M' の定義は

$$\mathsf{M}' = \cdots \mid \mathsf{D}_i'\,(\mathsf{Y}_1, \cdots, \mathsf{Y}_m, \mathsf{M}', \cdots, \mathsf{M}') \mid \cdots$$

となる．ここで D_i' はデータ構成子，Y_j は $t_j\,(d)$ の型であり，右辺の M' の数は l 個である．さらに，M' を定める関手は $\mathsf{M}_i' = !\mathsf{Y}_1 \times \cdots \times !\mathsf{Y}_m \times \mathsf{I} \times \cdots \times \mathsf{I}$（I は l 個）となる．

$t_j\,(d)$ は n に移動するので，$\mathtt{n} = \cdots + \mathtt{n}_i + \cdots$，$\mathtt{p}'' = \cdots \triangledown \mathtt{p}_i'' \triangledown \cdots$ は，下のように定めればよい．

$$\mathtt{n}_i\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j) = (t_j\,(\mathtt{d}), \mathtt{r}_j)$$

$$\mathtt{p}_i''\,(\mathtt{u}_j, \mathtt{r}_j) = \mathcal{E}_i'\,(\mathtt{v}_j, \mathtt{d}, \mathtt{r}_j) \text{ 中の } t_j\,(\mathtt{d}) \text{ を } \mathtt{u}_j \text{ に置換したもの} = \mathcal{E}_i''\,(\mathtt{u}_j, \mathtt{r}_j)$$

ここで，$\mathtt{u}_j$ は $t_j\,(\mathtt{d})$ に対応する引数である．

以上から $[\![\mathtt{p}', \mathtt{id}, \mathtt{q}]\!]_{\mathsf{M},\mathsf{M}}$ を $[\![\mathtt{p}'', \mathtt{id}, \mathtt{n} \circ \mathtt{q}]\!]_{\mathsf{M}',\mathsf{M}'}$ と変換できた．$\mathtt{q}' = \mathtt{n} \circ \mathtt{q}$ の定義は次のとおり．

$$\mathtt{q}'\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d}) = (i, (t_j\,(\mathtt{d}), \underline{(\mathtt{r}_j, \alpha_j\,\mathtt{d})}))$$

最後に，$\mathsf{M}_i'\,\mathtt{g} \circ \mathtt{q}' = \mathtt{q}'' \circ \mathtt{g}$ に従って g を抽出する．両辺に引数 $(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d})$ を与えると，

$$\text{左辺} = \mathsf{M}_i'\,\mathtt{g}\,(\mathtt{q}'\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d})) = (i, (t_j\,(\mathtt{d}), \underline{\mathtt{g}\,(\mathtt{r}_j, \alpha_j\,\mathtt{d})}))$$

$$\text{右辺} = \mathtt{q}''\,(\mathtt{g}\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d}))$$

ここで，高階 Cata のときの g を用いて $\mathtt{g}\,(\mathtt{x}, \mathtt{d}) = (\mathtt{x}, g\,\mathtt{d})$ と定義すると，

$$\text{左辺} = (i, (t_j\,(\mathtt{d}), \underline{(\mathtt{r}_j, g\,(\alpha_j\,\mathtt{d}))})) = (i, (t_j'\,(g\,\mathtt{d}), \underline{(\mathtt{r}_j, \beta_j\,(g\,\mathtt{d}))}))$$

$$\text{右辺} = \mathtt{q}_i''\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), g\,\mathtt{d})$$

となる．したがって，

$$\mathtt{q}''\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d}) = (i, (t_j'\,(\mathtt{d}), \underline{(\mathtt{r}_j, \beta_j\,\mathtt{d})}))$$

とおけば，

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

と変換されたことになる．

以上より，高階 Cata による変換で g を抽出することが可能ならば，Hylo による変換においても g を抽出することができることが示された．

5.4 最終結果の整理

以上で得られた結果を見やすい形に書き直す．

高階 Cata による変換の場合，$f'' = (\![p'']\!)_{\mathsf{F}}$ とおくと f'' の定義は次のとおり．

$$f''\,(\mathsf{C}_i\,(v_j, r_j)) = \lambda d\,.\,\mathcal{E}_i''\,(t_j'\,(d), \underline{f''\,r_j\,(\beta_j\,d)})$$

Hylo を用いた変換の場合，$\mathtt{f}'' = [\![\mathtt{p}'', \mathtt{id}, \mathtt{q}'']\!]_{\mathsf{M}',\mathsf{M}'}$ とおくと，$\mathtt{f}'' = (\![\mathtt{p}'']\!)_{\mathsf{M}'} \circ [\![\mathtt{q}'']\!]_{\mathsf{M}'}$ である．$[\![\mathtt{q}'']\!]_{\mathsf{M}'}$ と $(\![\mathtt{p}'']\!)_{\mathsf{M}'}$ の定義は次のとおり．

$$[\![\mathtt{q}'']\!]_{\mathsf{M}'}\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d}) = \mathsf{D}_i'\,(t_j'\,(d), \underline{[\![\mathtt{q}'']\!]_{\mathsf{M}'}\,(\mathtt{r}_j, \beta_j\,\mathtt{d})})$$

$$(\![\mathtt{p}'']\!)_{\mathsf{M}'}\,(\mathsf{D}_j\,(\mathtt{u}_j, \mathtt{r}_j)) = \mathcal{E}_i''\,(\mathtt{u}_j, \underline{(\![\mathtt{p}'']\!)_{\mathsf{M}'}\,\mathtt{r}_j})$$

したがって，

$$\begin{aligned}\mathtt{f}''\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d}) &= (\![\mathtt{p}'']\!)_{\mathsf{M}'}\,([\![\mathtt{q}'']\!]_{\mathsf{M}'}\,(\mathsf{C}_i\,(\mathtt{v}_j, \mathtt{r}_j), \mathtt{d})) \\ &= (\![\mathtt{p}'']\!)_{\mathsf{M}'}\,(\mathsf{D}_i'\,(t_j'\,(d), \underline{[\![\mathtt{q}'']\!]_{\mathsf{M}'}\,(\mathtt{r}_j, \beta_j\,\mathtt{d})})) \\ &= \mathcal{E}_i''\,(t_j'\,(\mathtt{d}), \underline{(\![\mathtt{p}'']\!)_{\mathsf{M}'}\,([\![\mathtt{q}'']\!]_{\mathsf{M}'}\,(\mathtt{r}_j, \beta_j\,\mathtt{d}))}) \\ &= \mathcal{E}_i''\,(t_j'\,(\mathtt{d}), \underline{\mathtt{f}''\,(\mathtt{r}_j, \beta_j\,\mathtt{d})})\end{aligned}$$

これで f'' と $\mathtt{f}''$ は実質的に同じ内容の定義を持つことが確かめられた．

6. おわりに

本論文では，蓄積引数を持つ関数を用いた関数プログラムの融合変換について，2 つのアプローチ ―― 高階関数の Catamorphism に基づく方法と，融合型を導入することによる Hylomorphism に基づく方法 ―― を用いた具体的手順を提示・検討し，前者に基づく融合変換操作は，後者による融合変換に帰着できることを示した．

構成的アルゴリズム論に関する研究の発展により，再帰パターンの “標準形” としての三つ組の Hylo 表現の有効性[4),6)]が明らかになってきている．本論文の結果は，その有効性をさらにサポートする事実としてとらえることができる．

両変換過程を見て分かるとおり，融合変換結果が有効であるか否かは，2 回目の融合変換定理の適用，すなわち，蓄積変数側にうまく関数を抽出できるかどうかに依存する．高階 Cata による変換では高階融合定理 2 が，Hylo による変換では Hylo 融合定理の 2 番目がその役割を担っており，この 2 つの規則は互いに

密接にかかわっていることが分かる．この点は，従来はほとんど指摘されていない点である．また従来の研究では，融合定理を逆に用いて関数を抽出することによって効率が良くなることがある，という点にはあまり目を向けられていない．これは一見融合変換とは相反するように見えるかもしれないが，たとえばHyloによる変換の場合，

$$[\![\mathtt{p}'', \mathtt{id}, \mathtt{q}']\!]_{\mathsf{M}',\mathsf{M}'}\ (\mathtt{x}, \mathtt{d}) = [\![\mathtt{p}'', \mathtt{id}, \mathtt{q}'']\!]_{\mathsf{M}',\mathsf{M}'}\ (\mathtt{x}, g\,\mathtt{d})$$

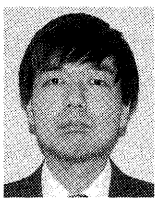
によって，蓄積する値を d の型のデータではなく g d の型のデータで表現でき，メモリを必要とするデータの生成を抑制することができるのである．

現在我々は，構成的アルゴリズム論に基づく関数プログラム変換システム HYLO Calculator[8] を開発中である．このシステムでは，今までに得られた様々な理論的成果の有効性を，実働システムの構築を通して確認していくことを目的としている．本論文における研究によって得られた知見が，今後，実用的なプログラム変換システムの構築に生かされることが期待される．

参考文献

1) Malcolm, G.: Data Structures and Program Transformation, *Science of Computer Programming*, Vol.14, No.2–3, pp.255–279 (1990).
2) Meijer, E., Fokkinga, M. and Paterson, R.: Functional Programming With Bananas, Lenses, Envelopes and Barbed Wire, *Proc. FPCA '91*, LNCS 523, pp.124–144, Springer-Verlag (1991).
3) Sheard, T. and Fegaras, L.: A Fold for All Seasons, *Proc. FPCA '93*, pp.233–242, ACM Press (1993).
4) Takano, A. and Meijer, E.: Shortcut Deforestation in Calculational Form, *Proc. FPCA '95*, pp. 306–313, ACM Press (1995).
5) Launchbury, J. and Sheard, T.: Warm Fusion: Deriving Build-Catas from Recursive Definitions, *Proc. FPCA '95*, pp.314–323, ACM Press (1995).
6) Hu, Z., Iwasaki, H. and Takeichi, M.: Deriving Structural Hylomorphisms from Recursive Definitions, *Proc. 1st ACM SIGPLAN Intl. Conf. on Functional Programming*, pp.73–82, ACM Press (1996).
7) Bird, R.: The Promotion and Accumulation Strategies in Transformational Programming, *ACM Trans. Prog. Lang. Syst.*, Vol.6, No.4, pp.487–504 (1984).
8) Onoue, Y., Hu, Z., Iwasaki, H. and Takeichi, M.: A Calculational Fusion System HYLO, *Proc. IFIP TC2 Working Conf. on Algorithmic Languages and Calculi*, pp.76–106, Chapman & Hall (1997).

(平成 9 年 9 月 3 日受付)
(平成 10 年 1 月 16 日採録)



岩崎 英哉（正会員）

1960 年生．1983 年東京大学工学部計数工学科卒業．1988 年東京大学大学院工学系研究科情報工学専攻博士課程修了．同年同大学計数工学科助手，1993 年同大学教育用計算機センター助教授を経て，1996 年東京農工大学工学部電子情報工学科助教授．工学博士．日本ソフトウェア科学会，ACM 各会員．

胡 振江（正会員）

1966 年生．1988 年中国上海交通大学計算機科学系を卒業．1996 年東京大学大学院工学系研究科情報工学専攻博士課程修了．同年学術振興会特別研究員を経て，1997 年東京大学大学院工学系研究科情報工学専攻助手，同年 10 月より同専攻講師．博士（工学）．日本ソフトウエア科学会，ACM 各会員．