ノート　Notes

プログラムの合成における修正，結合の情報を得る方法について

川　崎　治　夫*

A Method for Obtaining Information about Modification and Combination in Program Synthesis

HARUO KAWASAKI*

Synopsis: When we synthesize iterative list-processing programs by modification and combination, it is very difficult to obtain all information. We introduce basic relation and its concept. That information can be easily obtained at this basic relation level.

要旨：修正，結合を使って，反復型リスト処理プログラムを合成するときに，その修正，結合に関するすべての情報を自動的に得ることは非常に困難である。そこで，基本関係という概念を導入する。このレベルで考えると比較的簡単に，修正，結合の情報が得られる。

1. はじめに

先に反復型リスト処理プログラム合成のためのシステム [3] を提案したが，そこにおいては，システムに登録された既存のプログラムを修正，結合して新しいプログラムを得るという方法を採用している。このような方向については，まだあまり研究されておらず [2] このシステムを人工知能における問題として捉えてみた場合，様々な問題が生じてくるが，その中でも基本的な問題点として次のようなものが挙げられる。

1　どの程度の情報をシステムに与えておくか？

2　プログラムの，どこの部分を，どのように修正するか，ということに関する情報を，いかにして得るのか？

3　どのプログラムをベースにして修正，結合を実行すれば，目的のプログラムを得ることができるか？

これらは，どれも人工知能における問題またシステムの自動化という面からみても簡単な問題ではない。今報告においては，プログラムに対して基本関係という概念を定義し，1および2の解決策とする。基本関係を使っていくと，少なくとも修正，結合に関する最重要情報は得られているので，修正，結合に関するすべての情報は得られなくても，プログラムのどの部分

* 電子計算機センター　専任講師
Lecturer, Computer Center

を，どのような方向に修正していけばよいかという目標を得ることができる。

2. 作　用　素

プログラムの修正，結合には以下のような作用素を使うものとする。[3]

（ i ）　$\sigma'\theta_1, \theta_2, \ldots, \theta_n'$

作用素 σを，作用素による表現$\theta_1, \theta_2, \ldots, \theta_n$ によって変更し利用することを意味する。

（ ii ）　$x \leftarrow y$

y を x と名前を変えることを意味する。

（ iii ）　$m \prec l$

リスト m, l が与えられたとき，mの最後のセル i に l をつなぐことを意味する。

（ iv ）　$m \prec l$

リスト m, l が与えられたとき，mの最初のセルに l をつなぐことを意味する。

（ v ）　Nr

代入文，条件文の右辺の変更のために用いる。

（ vi ）　Nl

代入文，条件文の左辺の変更のために用いる。

（vii）　N

節点の部分に新たに機能を追加する。

（viii）　λl

特定の代入文をプログラムから削除する。

3. 修正，結合の情報を得る方法について

反復型リスト処理プログラム合成システムに与える情報は，既存のプログラムに対する修正，結合に関する情報である。しかし実際には，どのプログラムのどこを修正するか，あるいは，何と，どのプログラムを結合するかという情報を自動的に得るのは，大変困難なことである。しかしシステムに登録してあるプログラムに関する情報によっては，これらの情報の全部あるいは，ほとんどを得ることができる。ここでは，*RUNCOPY*，[3] がシステムに登録されているので，このプログラムを中心にして述べる。

3.1　基本関係

反復型リスト処理プログラム合成システムを自動化していくためには，まず得ようとするプログラムの入出力関係を明確化していく必要がある。そのために，このシステムでは何個かの

入出力の例を与える。たとえば，リスト $k(=nil)$ を最初から順にたどってコピーし，m という新たなリストを作るプログラム $RUNCOPY(k)$ を合成したいときには，次の3個の例をシステムに与えてみる。

$\{(a) \longrightarrow (a),$

$(ab) \longrightarrow (ab),$

$(abc) \longrightarrow (abc)\}$　（矢印の左側が入力リストで右側が出力リスト）

この3個の例に対して，まず cons,car,cdr を使用して左側のリスト構造から右側のリスト構造を得る関数を作成する。第 n 番目の例に対する関数を g_n とすると各々以下のようになる。

$g_1[x] = \text{cons}\ [\text{car}[x]; \text{NIL}]$

$g_2[x] = \text{cons}\ [\text{car}[x]; \text{cons}\ [\text{cadr}[x]; \text{NIL}]$

$g_3[x] = \text{cons}\ [\text{car}[x]; \text{cons}\ [\text{cadr}\ [x]; \text{cons}\ [\text{caddr}\ [x; \text{NIL}]]]$

ここまでは，Summers[4] が使用している方法と同じである。次に目的であるところの修正，結合の情報を得やすくするためにシステムは，次の append の性質に関する規則を使って g_2, g_3 を書き換える。

$$\text{append}\ [\text{cons}\ [x; \text{NIL}]; y] = \text{cons}\ [x; y] \qquad (R\text{–}1)$$

$$\text{append}\ [x; \text{append}\ [y; z]] = \text{append}\ [\text{append}\ [x; y]; z] \qquad (R\text{–}2)$$

すると上の関係は，次のようにあらわされる。

$g_1[x] = \text{cons}\ [\text{car}[x]; \text{NIL}]$

$g_2[x] = \text{append}\ [\text{cons}\ [\text{car}[x]; \text{NIL}]; \text{cons}\ [\text{cadr}[x]; \text{NIL}]]$

$g_3[x] = \text{append}\ [\text{append}\ [\text{cons}\ [\text{car}[x]; \text{cons}\ [\text{cadr}[x]; \text{NIL}]]; \text{cons}\ [\text{caddr}[x]; \text{NIL}]]$

これより $g_2[x] = \text{append}\ [g_1[x]; \text{cons}\ [\text{cadr}[x]; \text{NIL}]]$

$g_3[x] = \text{append}\ [g_2[x]; \text{cons}\ [\text{caddr}[x]; \text{NIL}]]$ となるからシステムは，次の関係を推論する。

$$\begin{cases} g_1[x] = \text{cons}\ [\text{car}[x]; \text{NIL}] \\ g_{n+1}[x] = \text{append}\ [g_n[x]; \text{cons}\ [\underbrace{\text{cadd}\cdots\text{dr}}_{n}[x]; \text{NIL}] \end{cases} \qquad (n = 1, 2, 3, \cdots)$$

これを $RUNCOPY$ の基本関係と呼ぶことにする。また上の推論に関しては，Summers [4] の differencing と同様にすれば得られる。

3.2 基本関係とプログラムの関係について

次に得られた基本関係とプログラム $RUNCOPY$ との関係について述べる。$RUNCOPY$ と

基本関係を比較してみると $g_1[x]$ は，*RUNCOPY* においては f_{23} に相当し，$g_{n+1}[x]$ は f_{67} に相当する。

この対応関係は，システムに情報として与えておく。また両者をくらべて変数の対応関係と状態変化も情報として登録しておく。以上をまとめると以下のようになる。

$$(m \xrightarrow{\alpha} nil,\ \hat{y} \xrightarrow{hd(j)} nil) \cdots\cdots g_n[x],\ \mathrm{cons}\,[\underbrace{\mathrm{cadd}\cdots\mathrm{dr}}_{n}[x];\ \mathrm{NIL}]$$

$$\Downarrow$$

$$(m \xrightarrow{\alpha} \hat{y} \xrightarrow{hd(j)} nil) \cdots\cdots \mathrm{append}\,[g_n[x];\ \mathrm{cons}\,[\underbrace{\mathrm{cadd}\cdots\mathrm{dr}}_{n}[x];\ \mathrm{NIL}]$$

$$\Downarrow$$

$$(m \xrightarrow{\alpha\cdot hd(j)} nil) \cdots\cdots g_{n+1}[x] = \mathrm{append}\,[g_n[x];\ \mathrm{cons}\,[\underbrace{\mathrm{cadd}\cdots\mathrm{dr}}_{n}[x];\ \mathrm{NIL}]]$$

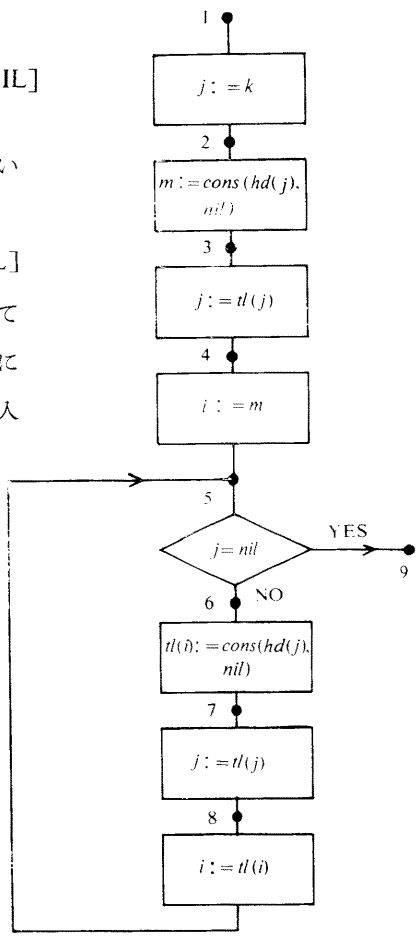
$$m \xrightarrow{hd(j)} nil \Leftrightarrow g_1[x]$$

$$m \xrightarrow{\alpha} nil \Leftrightarrow g_n[x]$$

$$\hat{y} \xrightarrow{hd(j)} nil \Leftrightarrow \mathrm{cons}\,[\underbrace{\mathrm{cadd}\cdots\mathrm{dr}}_{n}[x];\ \mathrm{NIL}]$$

尚 $\hat{y}$ は，y がプログラムの表面上には，あらわれていない変数であることを示す。

また状態変化は $g_n[x]$ と $\mathrm{cons}\,[\underbrace{\mathrm{cadd}\cdots\mathrm{dr}}_{n}[x];\ \mathrm{NIL}]$ が与えられて $g_{n+1}[x]$ が得られるまでの変化を示している。新たな反復型リスト処理プログラムを得るためには，まずそのプログラムが得られたと仮定したときの入出力例をいくつか与える。そして基本関係を得るのであるが，得られたものとシステムに登録している基本関係と比較して修正，結合に関する情報を得るのである。いくつかの入出力例を与えて，それからプログラムの構造を推定するというのは，プログラムの自動合成における一方法であるが，ここでは，それから得られる基本関係のレベルで目的のプログラムを得るための修正，結合の情報を得ていくのである。



(*RUNCOPY*(*k*))

$A = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$F = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}\}$

4. 実　　例

4.1 入力リストの最初のセルと同じ内容をすべてのセルがもつようなリストを作るプログラム

このプログラムの合成に対しては，次のような3個の例をシステムに与える。

{(a) ⟶ (a),

(ab) ⟶ (aa),

(abc) ⟶ (aaa)}

第 n 番目の例に対する関数を g_n とすると各々以下のように表現される。

$g_1[x]=\text{cons}\ [\text{car}\ [x]\ ;\ \text{NIL}]$

$g_2[x]=\text{cons}\ [\text{car}\ [x]\ ;\ \text{cons}\ [\text{car}\ [x]\ ;\ \text{NIL}]]$

$g_3[x]=\text{cons}\ [\text{car}\ [x]\ ;\ \text{cons}\ [\text{car}\ [x]\ ;\ \text{cons}\ [\text{car}\ [x]\ ;\ \text{NIL}]]]$

(R－1)，(R－2) を適用すると基本関係は以下のようになる。

$$\begin{cases} g_1[x]=\text{cons}\ [\text{car}[x]\ ;\ \text{NIL}] \\ g_{n+1}[x]=\text{append}[g_n[x]\ ;\ \text{cons}\ [\text{car}[x]\ ;\ \text{NIL}]] \qquad (n=1,\ 2,\ 3,\cdots) \end{cases}$$

これを *RUNCOPY* の基本関係とくらべると $g_{n+1}[x]$ において $\underbrace{\text{cadd}\cdots\text{dr}}_{n}[x]$ が car[x] にかわっている。変数の対応関係は $\hat{y}\xrightarrow{hd(j)}nil \Leftrightarrow \text{cons}\ [\underbrace{\text{cadd}\cdots\text{dr}}_{n}[x];\ \text{NIL}]$ だから $hd(j)$ を修正する必要が生じていることがわかる。ここで x は入力リストを意味するので f_{67} において $hd(j)$ を $hd(k)$ に修正すればよいことがわかる。したがってこれらは，作用素を使って以下のようにあらわされる。

$$\begin{cases} if \quad k=nil \longrightarrow nil \\ \sim \qquad\quad \longrightarrow RUNCOPY(k)\ `Nr(f_{67})=(hd(j),\ hd(k))` \end{cases}$$

4.2 リストを逆順にするプログラム（セルをコピーしながら逆順にする）

このプログラムの合成に対しては，次のような3個の例をシステムに与える。

{(a)⟶(a),

(ab)⟶(ba),

(abc)⟶(cba)}

第 n 番目の例に対する関数を g_n とすると各々以下のようになる。

$g_1[x]=\text{cons}\ [\text{car}\ [x]\ ;\ \text{NIL}]$

$g_2[x]=\text{cons}\ [\text{cadr}\ [x]\ ;\ \text{cons}\ [\text{car}\ [x]\ ;\ \text{NIL}]]$

$g_3[x]=\text{cons}\ [\text{caddr}\ [x]\ ;\ \text{cons}\ [\text{cadr}\ [x]\ ;\ \text{cons}\ [\text{car}\ [x]\ ;\ \text{NIL}]]]$

(R－1)，(R－2) を適用して以下のような基本関係を得る。

$$\begin{cases} g_1[x] = \text{cons}\ [\text{car}[x]\ ;\ \text{NIL}] \\ g_{n+1}[x] = \text{append}\ [\text{cons}\ [\underbrace{\text{cadd}\cdots\text{dr}}_{n}\ [x]\ ;\ \text{NIL}]\ ;\ g_n[x]] \qquad (n = 1, 2, 3, \cdots) \end{cases}$$

これを *RUNCOPY* の基本関係式とくらべると $g_{n+1}[x]$ において $\text{cons}[\underbrace{\text{cadd}\cdots\text{dr}}_{n}[x]; \text{NIL}]$ と $g_n[x]$ の位置が交換されている。したがって状態変化の方を検討してみる必要がでてくる。

① $(\hat{y} \xrightarrow{hd(j)} nil,\ m \xrightarrow{\alpha} nil) \cdots\cdots \text{cons}\ [\underbrace{\text{cadd}\cdots\text{dr}}_{n}\ [x]\ ;\ \text{NIL}],\ g_n\ [x]$

② $(\hat{y} \xrightarrow{hd(j)} m \xrightarrow{\alpha} nil) \cdots\cdots\cdots\cdots \text{append}\ [\text{cons}\ [\underbrace{\text{cadd}\cdots\text{dr}}_{n}\ [x]\ ;\ \text{NIL}]\ ;\ g_n[x]]$

③ $(\hat{y} \xrightarrow{hd(j)\cdot\alpha} nil) \cdots\cdots\cdots g_{n+1}[x] = \text{append}\ [\text{cons}\ [\underbrace{\text{cadd}\cdots\text{dr}}_{n}\ [x]\ ;\ \text{NIL}]\ ;\ g_n[x]]$

②において $\hat{y} \xrightarrow{hd(j)} m \xrightarrow{\alpha} nil$ というリスト $\hat{y}$ ができたので，まず①の段階で y を表面上にだして　　$y \longleftarrow cons(hd(j),\ nil)$ ……イ　　を追加する必要がある。

さらに②より　　$y \succ m$ ……ロ　　を得る。また③よりこのままにしておくとできたリストに y という名前がつくので

$m \longleftarrow y$ ……ハ　　となる。ロとハを合せて

$m \leftarrow y \succ m$ ……ニ　　と表現できる。

f_{67} は $tl(i) := cons(hd(j),\ nil)$ だからイより $Nl(f_{67}) = (tl(i),\ y)$ を得る。

またニより $N(7) = m \longleftarrow y \succ m$ を得る。現在のところまだ基本関係からリストの最後のセルを指す i に関する情報を得る方法は得られていないが，リストを逆順にするときには，セルをリストの後方につなぐことはないので，$\Lambda\ (f_{45},\ f_{85})$ を得るのは難しいことではないと考えられる。したがってこれらは作用素を使って以下のように表現される。

$$if \quad k = nil \longrightarrow nil$$

$$\sim \quad \longrightarrow RUNCOPY\ `Nl(f_{68}) = (tl\ (i),\ y),$$

$$N(7) = m \leftarrow y \succ m,\ \Lambda(f_{45}, f_{85})'$$

4.3　入力リストを倍にコピーして得られるリストを作るプログラム

このプログラムの合成に対しては，次のような3個の例をシステムに与える。

{(a) ⟶ (aa),
(ab) ⟶ (aabb),
(abc) ⟶ (aabbcc)}

第 n 番目の例に対する関数を g_n とすると以下のようになる。

$g_1[x] = \text{cons}\ [\text{car}\ [x]\ ;\ \text{cons}\ [\text{car}[x]\ ;\ \text{NIL}]$

$g_2[x] = \text{cons}\ [\text{car}\ [x]\ ;\ \text{cons}\ [\text{car}[x]\ ;\ \text{cons}\ [\text{cadr}\ [x]\ ;$
$\text{cons}\ [\text{cadr}\ [x]\ ;\ \text{NIL}]]]]$

$$g_3[x] = \text{cons}\ [\text{car}\ [x]\ ;\ \text{cons}\ [\text{car}\ [x]\ ;\ \text{cons}\ [\text{cadr}\ [x]\ ;$$
$$\text{cons}\ [\text{cadr}\ [x]\ ;\ \text{cons}\ [\text{caddr}\ [x]\ ;\ \text{cons}\ [\text{caddr}\ [x]\ ;\ \text{NIL}]]]]]]$$

(R－1)，(R－2) を適用して以下のような基本関係を得る。

$$\begin{cases} g_1[x] = \text{append}\ [\text{cons}\ [\text{car}\ [x]\ ;\ \text{NIL}]\ ;\ \text{cons}\ [\text{car}\ [x]\ ;\ \text{NIL}]] \\ g_{n+1}[x] = \text{append}\ [\text{append}\ [g_n[x]\ ;\ \text{cons}\ [\underbrace{\text{cadd}\cdots\text{dr}}_{n}\ [x]\ ;\ \text{NIL}]]\ ; \\ \qquad\qquad \text{cons}\ [\underbrace{\text{cadd}\cdots\text{dr}}_{n}\ [x]\ ;\ \text{NIL}]] \qquad (n=1, 2, 3,\ldots) \end{cases}$$

RUNCOPY の基本関係とくらべると $g_1[x]$ において cons [car[x]; NIL] が，$g_{n+1}[x]$ において cons [$\underbrace{\text{cadd}\cdots\text{dr}}_{n}$ [x]; NIL] が追加されている。したがってこの分を情報として追加すればよいことになる。また cons [car [x]; NIL] と cons [$\underbrace{\text{cadd}\cdots\text{dr}}_{n}$ [x]; NIL] は各々 $g_1[x]$ と $g_{n+1}[x]$ で既に与えられている情報である。まず $g_1[x]$ より f_{23} の右辺 $cons\,(hd(j),\ nil)$ と同じものを m につなぐ必要がある。よってまず

$$tl(m) \longleftarrow cons\,(hd(j),\ nil) \quad \cdots\cdots イ$$

を得る。このとき i で最後のセルを指すために

$$i \longleftarrow tl(m) \quad \cdots\cdots ロ$$

を得る。

次に $g_{n+1}[x]$ より f_{67} の右辺 $cons\,(hd(j),\ nil)$ と同じものを m につなぐ必要がでてくる。

このときは，i で最後のセルを指しているので

$$tl(i) \longleftarrow cons\,(hd(j),\ nil) \quad \cdots\cdots ハ$$

を得る。

しかし f_{67} で $tl(i) := cons\,(hd(j),\ nil)$ がでているので i をセル1個分進めておく必要が生じ，ハの前に

$$i \longleftarrow tl(i) \quad \cdots\cdots ニ$$

が必要となる。

f_{45} は $i := m$ だから ロについては，$N_r(f_{45}) = (m,\ tl(m))$ とすればよい。したがってこれらは作用素を使って以下のように表現される。

$$\begin{aligned} &if \quad k = nil \longrightarrow nil \\ &\sim \qquad \longrightarrow RUNCOPY(k)\text{‘}\ N(3) = tl(m) \longleftarrow cons\,(hd(j),\ nil), \\ &\qquad\qquad N_r(f_{45}) = (m,\ tl(m)), \\ &\qquad\qquad N(7) = i \longleftarrow tl(i), \\ &\qquad\qquad N(7) = tl(i) \longleftarrow cons\,(hd(j),\ nil)\text{’} \end{aligned}$$

5. おわりに

基本関係は，プログラムの最重要部を表現しているので，修正，結合の情報を得るには最適であると考えられる。今後この方式の適用範囲をいかに広げていくかということが重要な問題となるであろう。また，システムに登録されているどのプログラムを基礎にしてプログラムを

作っていくかという情報は，入力時にシステムに与えるいくつかの入出力例のもつ特徴をくらべることによって情報を得られるのではないかと考えている。

(1982年2月6日受理)

文　献

1) Burstall, R. M.: Some techniques for proving correctness of programs which alter data structures, Machine Intelligence, vol. 7 (1972), 23-50.
2) Dershowitz, N. and Manna, Z.: The evolution of programs: A system for automatic program modification, IEEE Trans. Software Engineering, SE-3 (1977), 377-385
3) 川崎治夫：反復型リスト処理プログラムの実用的合成について，国士舘大学電子計算機センター紀要，第2号 (1981)，1-13
4) Summers, P. D.: A methodology for LISP program construction from examples, J. ACM, 24 (1975), 161-175