数理解析研究所講究録
1125巻 2000年 162-171

集合表現を含む仕様からのルール生成

Rule Generation from Specification That Consist of Set Representation

辻 武士[†] Takeshi TSUJI　赤間 清[†] Kiyoshi AKAMA　宮本 衛市[†] Eiichi MIYAMOTO

[†]北海道大学 システム情報工学専攻
Division of Systems and Information Engineering,
Hokkaido University

概 要

与えられた問題を解決するプログラムを自動的に生成する方法は，問題を高速に解決するために重要である．プログラムを自動生成するためには，問題を正当性を定義できる形で定式化する必要がある．また，定式化された問題からそれを解く計算方法を得るための理論が必要である．本論文では，$S = \{Y|q(X,Y)\}$ のように X に依存した条件 $q(X,Y)$ によって集合 S が決定される時，X からその集合 S を求めるプログラムを生成する方法を提案する．プログラムを自動的に生成するためには，問題に十分な理論を与えられる表現が必要となる．本論文では，等価変換パラダイムにおいて $q(X,Y)$ と S の関係を記述するために，**setof参照**を導入する．setof参照を用いた問題記述からのプログラム生成は，この問題を正しく計算する**等価変換ルール**の生成として捉えられる．本論文のルール生成によって，条件を満たす項の集合を効率的に計算するプログラムを自動的に生成することができる．

1 はじめに

本論文の目的は，与えられた条件を満たす集合を計算するプログラムを自動的に生成するための方法の基礎を検討することである．

プログラムを生成するためには，プログラムによって計算される問題を理論的に定式化する必要がある．宣言的記述の等価変換による問題記述の中で,「集合がある条件を満たしている」という 関係は，setof参照を用いて表現できる [1]．setof参照は，等価変換モデルにおいて，宣言的記述の構成要素である**参照**の特殊形として理論化できる．また，setof参照を含む宣言的記述の意味を保存したままsetof参照を書き換える等価変換ルールが用意されている．

本論文では，集合を計算するプログラム生成を，setof参照を用いた問題の定義からのルール生成として扱い，setof参照を効率的に計算するルールを生成する．

setof参照を変換する基本的な変換ルールは，既に存在する [1]．本論文では setof 参照の定義からルール生成の基礎となるメタルールを作る。そしてこのメタルールによって、setof参照を用いた問題の仕様から条件を用いて表される集合の計算プログラムを生成する．

本論文の構成は以下のようになる．初めにプログラムを生成する問題を定義する．次に，ルール生成の基本的な枠組となるメタルールについて解説する．最後に，setof参照を計算するルールの生成を，その過程を示しながら行う．

2 プログラム生成の定式化

2.1 プログラム生成問題の定義

本論文では，ある2引数の述語 q が与えられた時，任意の入力 X に対して

$S(X) = \{Y|q(X,Y)\}$

を満たす集合 $S(X)$ を求めるプログラム P を生成する問題を考える．

例として，q が次のような関係を表す述語 $seg(X,Y)$ である場合を考える．

「Y は，与えられた整数列 X の，隣り合った要素のみからなる，空でない部分列である」

例えば，X が整数列 [2,1,5] の場合，$seg(X,Y)$ を満たす Y は6つ存在し，

[2],[1],[5],[2,1],[1,5],[2,1,5]

である．このときのプログラム生成の問題を特に **segs 問題**と呼ぶ．

segs **問題:**

「任意の入力 X に対して，条件 $seg(X,Y)$ を満たす Y 全体の集合 S を求めるプログラム P を求めよ」

$find\ P$

$such\ that\ P(X) = S = \{Y|seg(X,Y)\}$

この問題の解 P は，特定のプログラム言語による手続きである．

2.2 関数型言語によるプログラム生成

本節では P として許されるプログラム P の範囲を，関数プログラムに限定した場合を考える．この時 *segs* 問題は，

$segs(X) = \{Y|seg(X,Y)\}$

を満たす関数 $segs(X)$ の関数型言語による定義を得る問題だと考えることができる．その解は例えば次のようになる．

$$segs(X) \stackrel{\text{def}}{=} if\ x = [\,]\ then\ [\,]$$
$$else\ append(inits(X),\ segs(cdr(X)))$$
$$inits(X) \stackrel{\text{def}}{=} if\ x = [\,]\ then\ [\,]$$
$$else\ cons([car(X)],$$
$$distr(car(X),\ inits(cdr(X))))$$
$$distr(A,B) = if\ B = [\,]\ then\ [\,]$$
$$else\ cons([A,car(B)],\ distr(A,\ cdr(B)))$$

である．ただし，append はリストの連結を表す関数である．

segs 問題の解を関数型言語の手続き (関数定義) として自動的に得るためには，手続きによって解決すべき問題の定義を関数型言語で扱える形で表現できなければならない．しかし，関数型言語には条件を用いて集合を表すような記述は存在しない．このため，*segs* 問題を問題定義からのプログラム生成問題として定式化することができない．

2.3 論理型言語によるプログラム生成

本節では P として許されているプログラムの範囲を，論理プログラム (確定節の集合) に限定した場合を考える．この時 *segs* 問題は，与えられた整数列 X と求めたい集合 S が

$S = \{Y|seg(X,Y)\}$

という関係にあることを示す述語 $segs(X,S)$ の論理型言語 (節集合) による定義を求める問題だと考えることができる．

このとき，次のような解が考えられる．

$$segs([\,],[\,]) \leftarrow .$$
$$segs([A|R],S) \leftarrow inits([A|R],M),$$
$$segs(R,N),$$
$$append(M,N,S).$$
$$inits([\,],[\,]) \leftarrow .$$
$$inits([A|R],[[A]|Z]) \leftarrow inits(R,M),$$
$$distr(A,M,Z).$$
$$distr(A,[\,],[\,]) \leftarrow .$$
$$distr([X|R],[[A,X],Z]) \leftarrow distr(A,R,Z).$$

segs 問題を論理プログラムのプログラム変換によって自動的に解決するためには，整数列 X から集合 S を求める問題を論理プログラムで表現する必要がある．しかし純粋な論理プログラムを構成する確定節集合には，条件を用いて集合を表現するような記述方法が存在しない．

2.4 ビルトイン述語 bagof で拡張された論理型言語

論理型言語には集合を求めるための *bagof* 述語のようなビルトインが存在する．*bagof* 述語を用いると，*segs* 問題で X と S の関係を

$segs(X,S) \leftarrow bagof(Y,seg(X,Y),S).$

のように，集合の定義と同じ形で記述できて，これがそのまま *segs* を求めるプログラム P となる．

bagof 述語は条件を満たす集合を計算する問題を簡単に記述することができる．しかし *bagof* 述語は，論理型プログラムにおいて要素が条件で表さ

れる集合を計算する方法を提供するビルトインであり，論理式のように問題を理論的に定式化するものではない．このため，$bagof$ 述語で集合表現を含む問題を定式化しても，そこから論理型プログラムの変換によって集合を計算する効率的な論理プログラムを生成することができない．

2.5 等価変換モデルにおける segs 問題の定式化

本論文で採用するプログラム生成方法は，条件によって決定される集合に理論的な根拠を持った表現を与える．

本論文では，等価変換パラダイムにおけるルール生成問題の定式化を用いる．まず，$segs$ 問題の S の定義を

$$segs(X,S) \leftarrow setof(S,X,seg(X,Y)).$$

のような拡張された**確定節**で表現する．そして，P として許されるプログラムの範囲を，$segs$ アトムを計算する方法 (**等価変換ルール**，あるいは単にルール) の集合に限定する．

等価変換パラダイムにおける segs 問題の解は，次のようなルールの集合として与えられる．

$$segs([\,],S) \rightarrow [S=[\,]]$$

$$segs([A|R],S)$$
$$\rightarrow segs(R,S1),$$
$$p([A|R],S2),$$
$$append(S1,S2,S)$$

$$p([A],S) \rightarrow [S=[[A]]]$$

$$p([A,B|R],S)$$
$$\rightarrow [S=[[A]|S1]],$$
$$p(R,S2),$$
$$distr(A,S2,S1)$$

アトム $append(X,Y,Z)$ はリスト X と Y を連結したリストが Z であることを表し，アトム $distr(X,Y,Z)$ はリストのリスト Y の各要素リストの先頭に X を付加したリストが Z であることを表す．

問題の定義に用いる $setof(S,X,seg(X,Y))$ のような式は，等価変換パラダイムでは **setof 参照**として理論づけられている．setof 参照には，条件から集合を求める (効率の悪い) プログラムが，その理論づけに基づいた複数のルールの集合で与えられている．ルール生成ではこれらのルールをルール生成の元となる**メタルール**として，より効率のよいルールを自動的に生成することができる．

3 メタ節の変換によるルール生成

3.1 宣言的記述からのルール生成

等価変換パラダイムでは，問題を宣言的記述で表現し，等価変換ルールで変換する事によって問題を解決する．

従来の等価変換では，宣言的記述の意味を保存する等価変換ルールを，人間が直接作っていた．setof 参照は，集合の要素を，条件を用いて自然に表現する記述であるが，setof 参照の基本的な等価変換ルールだけによる計算は一般に非効率的である．ルール生成では，基本的なルールをルール生成の元となる**メタルール**として，より効率の良いルールを自動的に生成する．

等価変換パラダイムによる問題解決では，問合せの確定節を含む宣言的記述が入力であり，ルールによって変換された結果の宣言的記述が出力である．

これに対して本論文で行うルール生成は，**メタ問題記述**と呼ばれる複数の宣言的記述を代表するパターンで問題を定義し，複数のルールを代表するパターンであるメタルールによって変換して，変換前後のメタ問題記述からルールを生成する．

3.2 メタアトム，メタ節

宣言的記述 Q の確定節 P に幾つかの等価変換ルールを施した結果を P'とする．この時，P を P' へ書き換えるルールもまた等価変換ルールである．しかし，このような方法では，P を P'に変換するような，個々の問題に依存したルールしか生成できない．このため，ルール生成に使用する，複数のアトムを代表するパターンを用意する必要がある．

本論文では，ルール生成に&変数と#変数という変数を用いる．宣言的記述で使用される項の変数の代わりに&変数と#変数を用いた項をメタ項と呼ぶ．等価変換による問題解決で使用される項の変数は，メタ項には含まれない．&変数には任意の項が代入できる．また，#変数には任意の変数が代入できる．

述語名とメタ項の並びからなる式をメタアトムと呼ぶ．メタアトムは例えば $segs([\&A|\&R],\#Y)$ のような形をしている．

メタアトムを用いて，ルール生成の入力となるメタ節を定義する．メタアトムによって構成される確定節がメタ節である．

3.3 メタルール

ルール生成では，メタ節を変換してルールを生成するためのメタルールが必要となる．

メタルールの記述では，定数の他に *変数や %変数を用いる．通常の変数や &変数，#変数は用いない．*変数は “*” で始まる変数であり，任意のメタ項を代入できる．%変数は “%” で始まる変数であり，任意の #変数を代入できる．

メタルールは，書き換え対象であるメタ節のボディに，条件を満たすアトムの 1 個以上の列が存在する時に適用される．そして，それらを空列になおしたり ($\langle true \rangle$ で示す)，節を消去したり ($\langle false \rangle$ で示す)，アトムの 1 個以上の列に置き換えたりする．メタルールの具体的な例は，3.4節以降で説明する．

3.4 setof参照を変換するためのメタルール

本論文では，条件を満たす項の集合を setof 参照を用いて表し，setof 参照をメタ変換するメタルールによって集合を求めるプログラムを生成する．このために，setof 参照を変換するメタルールを用意する必要が屡．

setof 参照を含むメタ節の変換ルールとして，次のメタルールを定義する．これらのメタルールは，それぞれが対応するルールを持ち，その正当性を証明する事ができる [1].

setof 参照をメタ変換するメタルールは，本論文の例題に挙げた segs 問題や segs2 問題に限らず，setof 参照を用いて定義された集合を求める問題を解くプログラムの生成に使用できる．

【条件を宣言的記述に繰り込むメタルール】

このメタルールは，setof 参照が参照している宣言的記述に，条件の問合せ節を繰り込むためのメタルールである．

メタルールは次のような形になる．

$\langle *S, f_{*x,a}, Q \rangle \Leftrightarrow \langle *S, f_{*x,\phi(a)}, Q \cup Q' \rangle$

ただし，$Q' = \{\phi(a) \leftarrow a.\}$

変換前の setof 参照が参照する宣言的記述 (参照記述と呼ぶ)Q はアトム a を表す述語の正しい関係を定義している．ϕはアトムからアトムを得る関数で，$\phi(a)$ はアトム a の引数はそのままで述語名が新しくなったアトムを表す．このメタルールは，問合せに当たる節を参照記述に繰り込み，参照記述を具体的な問題を表す宣言的記述にする．

実際の計算では，setof 参照を含む宣言的記述と参照記述は，それぞれがワールドと呼ばれる計算空間に割り当てられて，ワールド間で情報を伝播しながら計算される．

【集合の要素を追加するメタルール】

このメタルールは，参照記述の中に解 (単位節が解に当たる) が現れた時，その解を集合の要素に追加する．

メタルールの記述は，次のようになる．

$\langle *S, f_{*X,\phi(a)}, Q \cup Q_a \rangle$

$\Leftrightarrow \langle \%S', f_{*X,\phi(a)}, Q \rangle, [*S = [x\sigma | \%S']]$

ただし，$Q_a = \{\phi(a\sigma) \leftarrow .\}$

ここでσは代入を表す．また，x は Q_aのアトム $\phi(a)$ の引数である．$x\sigma$は，参照記述が表す問題の解$\phi(a\sigma)$ に対応する集合$*S$ の要素を表している．

【空集合を導くメタルール】

このメタルールは，条件を満たすような項が存在しない時に空集合を導く．

メタルールは次のようになる．

$\langle *S, f_{*X,\phi(a)}, Q \cup \{\} \rangle \Leftrightarrow [*S = []]$

参照記述に$\phi(a)$ をヘッドとする節が一つもない時，即ち宣言的記述にこれ以上解が存在しない時に適用される．

【参照記述を分離するメタルール】

このメタルールは，setof 参照の参照記述のうち，$\phi(a)$ をヘッドに持つ節の集合 Q_{ans}に 2 つ以上の節が存在する場合に適用される．Q_{ans}の節を 1 個取り出した新しい宣言的記述 C とその残りの宣言的記述 Q'_{ans}に分け，それぞれを参照記述に持つ 2 つの setof 参照と，その集合の和集合を表す append アトムに展開する．Q をヘッドが$\phi(a)$ でない節の集合，C をある$\phi(a)$ 節とする．

メタルールは次のようになる．

$\langle *S, f_{*X,\phi(a)}, Q \cup Q_{ans} \rangle$

$\Leftrightarrow \langle \%S1, f_{*X,\phi(a)}, Q \cup \{C\} \rangle,$

$\langle \%S2, f_{*X,\phi(a)}, Q \cup Q'_{ans} \rangle,$

$append(\%S1, \%S2, *S)$

ただし，

$Q_{ans} = \{C\} \cup Q'_{ans} \quad (Q'_{ans} \neq \emptyset)$

【参照関数を書き換えるメタルール】

setof 参照の参照関数 $f_{X,\phi(a)}$は，参照記述の意味から，$\phi(a) \leftarrow .$の形の単位節の引数 Y を集めた集合を与える．この関数は，参照記述から X の集合

を得るものなので，この集合が正しく得られるように参照関数や参照記述を書き換えても，問題の意味は保存される．参照記述のうち，$\phi(a)$ アトムをヘッドとする節集合を Q_{ans}，Q_{ans}のすべての節のヘッドを$\phi(b)$ に変えた節集合を Q'_{ans}とする．ただし，アトム b の引数には，X が含まれていなければならない．この書き換えによって X を要素とする集合は変化しない．

メタルールは次のようになる．

$\langle *S, f_{*X,\phi(a)}, Q \cup Q_{ans}\rangle,$
$\Leftrightarrow \langle *S, f_{*X,\phi(b)}, Q_{seg} \cup Q'_{ans}\rangle$

【共通の解をくくり出すメタルール】

setof 参照 $\langle S, f_{X,\phi(a)}, Q \cup Q_{ans}\rangle$ の参照記述のうち，Q_{ans}は$\phi(a)$ をヘッドとする節集合を表すとする．Q_{ans}のすべてのヘッドの X にあたる引数，即ち参照される問題の解が，$[\&A|R]$ の形をしたリストであるとする．ただし，$\&A$ は，定数とする．

この時 S は，Rの部分だけを解とする新たな集合 S'の各要素に，A を付け加えた集合と等価である．

よって，次のようなメタルールが成立する．Q_{ans} の全ての節のヘッドの$*X$ にあたる引数が $[*A|*R]$ である時，それらの引数を$*R$ に書き換えた節集合を Q'_{ans}とすると，

$\langle *S, f_{*X,\phi(a)}, Q \cup Q_{ans}\rangle$
$\Leftrightarrow \langle \%S', f_{X,\phi(a)}, Q \cup Q'_{ans}\rangle,$
$\quad distr(A, \%S', *S)$

ここで，$distr(X,Y,Z)$ は，“リスト Y の各要素リストの先頭に X を加えたリストは Z である”という意味を持った述語である．$distr$ は基本的な述語であり，その変換ルールは既に用意されているものとする．

3.5 setof 参照を含む問題記述からのルール生成の特長

ルール生成は一般性が高く，論理プログラムの問題クラス (任意の Horn 節の集合全体) の全ての問題についてルール生成を行うことができる．本論文で提案する setof 参照を含む仕様からのルール生成は，ルール生成によってプログラムを自動生成できる問題のクラスをさらに広げることができる．また，ルール生成によって生成されたルールはモジュラー性が高く，複数の問題による再利用が用意である．

4 segs 問題を解くルールの生成

4.1 ルール生成の流れ

本章ではルール生成によって $segs$ 問題を解決する．問題解決の流れは以下のようになる．

1. 述語 $segs$ の定義を $setof$ 参照を含む宣言的記述で表現する．
2. $setof$ 参照やその他の述語を含んだメタ節を変換するためのメタルールを用意する．
3. $segs$ アトムを変換するルールを，上記のメタルールを用いて生成する．
4. $segs$ アトムの変換に伴って出現するアトムを処理するルールを生成する (補助ルールと呼ぶ)．
5. 生成したルールと補助ルールを合わせて，P を得る．

4.2 setof 参照を用いた $segs$ 問題の表現

部分列問題を setof 参照を含んだ宣言的記述で表現すると，次のようになる．

$Q_{segs} = \{$
$\quad segs(X,S) \leftarrow \langle S, f_{Y,seg(X,Y)}, Q_{seg}\rangle$
$\}$
ただし，
$Q_{seg} = \{$
$\quad seg(X,Y) \leftarrow append(M,B,X),$
$\qquad\qquad\quad append(A,Y,M),$
$\qquad\qquad\quad [Y \neq []].$
$\}$

ここで，$\langle S, f_{X,seg(X,S)}, Q_{seg}\rangle$ は「$seg(X,S)$ を満たす X の集合が S である」という意味の setof 参照である．

この宣言的記述を元にメタルールを準備し，$segs$ アトムを変換するルールを生成する．

4.3 その他のメタルール

3.4節で挙げた setof 参照のルールは，問題の具体的な形によらないメタルールである．segs 問題を解くために，segs の宣言的記述から得られるメタルールや他の基本的なルールを生成するためのメタルールを用意する．

【segs の宣言的記述から得られるメタルール】

r-1:

$segs(*X, *S) \Leftrightarrow \langle *S, f_{\%Y,seg(*X,\%Y)}, Q_{seg}\rangle$

$Q_{seg} = \{$
$seg(\%K, \%L) \leftarrow append(\%M, \%B, \%K),$
$append(\%A, \%L, \%M),$
$[\%L \neq []].$
$\}$

【seg の宣言的記述から得られるメタルール】

r-2:

$seg(*X, *Y) \Leftrightarrow append(\%M, \%B, *X),$
$append(\%A, *Y, \%M),$
$[*Y \neq []]$

【append のメタルール】

r-3:

$append(*X, *Y, *Z)$
$\Leftrightarrow [*X = []], [*Y = *Z]$
$\Leftrightarrow [*X = [\%A|\%R]], [*Z = [\%A|\%W]],$
$append(\%R, *Y, \%W)$

4.4 ルール生成の過程

以上で用意したメタルールを元に，部分列問題を解決する等価変換ルールを生成する．

4.4.1 $segs(\&X, \&S)$ からのルール生成

segs の第 1 引数を & 変数としたメタ節からルールを生成する．生成されたルールは，第 1 引数がどのような形でも適用できる．

c1: $h \leftarrow segs(\&X, \&S).$

segs アトムを展開 → setof 参照の繰り込み → 参照記述の seg アトムを展開

のように変換して，次のようなメタ節が得られる．

c4:

$h \leftarrow \langle \&S, f_{\#Y, \phi(seg(\&X, \#Y))}, Q_{seg} \cup Q_{ans2} \rangle.$
$Q_{seg} = \{$
$seg(\#K, \#L) \leftarrow append(\#M, \#D, \#K),$
$append(\#C, \#L, \#M),$
$[\#L \neq []].$
$\}$
$Q_{ans2} = \{$
$\phi(seg(\&X, \#Y)) \leftarrow append(\#M, \#B, \&X),$
$append(\#A, \#Y, \#M),$
$[\#Y \neq []].$
$\}$

これ以上はメタルールを適用できない．よって，次のルールができる．

ルール 1 *segs:0*

$segs(\&X, \&S) \rightarrow \langle \&S, f_{\#Y, seg(\&X, \#Y)}, Q_{seg} \cup Q_{ans} \rangle$

ただし，

$Q_{seg} = \{$
$seg(\#K, \#L) \leftarrow append(\#M, \#B, \#K),$
$append(\#A, \#L, \#M),$
$[\#L \neq []].$
$\}$
$Q_{ans} = \{$
$\phi(seg(\&X, \#Y)) \leftarrow append(\#M, \#B, \&X),$
$append(\#A, \#Y, \#M),$
$[\#Y \neq []].$
$\}$

このルールはアトムを左辺から右辺へ展開するルールであると同時に，右辺から左辺への畳み込みを行うルールでもある．また，生成したルールから新たなメタルールを作ることができる．ルールの & 変数を *変数に，# 変数を % 変数に変えたものは，メタルールとしてルール生成に利用できる．

4.4.2 $segs([], \&S)$ からのルール生成

次に，ルールの適用範囲を狭くして，segs の第 1 引数を空リストにしたメタ節にメタルールを適用する．適用範囲を狭くすることで，メタルールによる変換が進むことが期待される．

c1: $h \leftarrow segs([], \&S).$

segs アトムを展開 → setof の繰り込み変換を適用 → seg アトムを展開 → append アトムを展開 → 等式制約を解消 → append アトムを展開 →等式制約を解消 → 不等式制約を解消 → setof の消去変換を適用

といった変換により，次のメタ節が得られる．

c12: $h \leftarrow [\&S = []].$

以上より，次の等価変換ルールが得られる．

ルール 2 *segs:nil*

$segs([], \&S) \rightarrow [\&S = []]$

このルールを用いると，入力が空リストの場合，setof 参照の繰り込みなどの操作を介さずに，直接 $\&S$ を求められる．

4.4.3 $segs([\&A|\&R], \&S)$ からのルール生成

segs の第 1 引数を &変数の cons で与えたメタ節を変換する．

c1:$h \leftarrow segs([\&A|\&R], \&S).$

segs アトムを展開 → setof 参照の繰り込み変換を適用 → seg アトムを展開→ append アトムを展開 → 等式制約 3 を解消 → 不等式制約を解消 → append アトムを展開 → 等式制約 3 を解消 → 不等式制約を解消 → setof 参照の分解 → setof 参照の条件の書き換え → setof 参照の畳み込み

といった変換で，次のメタ節が得られる．

c17:

$h \leftarrow segs(\&R, \#S1),$
$\langle \#S2, f_{\#Y,\phi(seg([\&A|\&R],\#Y))}, Q_{seg} \cup QB_{ans} \rangle,$
$append(\#S1, \#S2, \&S).$
$Q_{seg} = \{$
$seg(\#K, \#L) \leftarrow append(\#M, \#B, \#K),$
$append(\#A, \#L, \#M),$
$[\#L \neq []].$
$\}$
$QB_{ans} = \{$
$\phi(seg([\&A|\&R], [\&A|\#E])) \leftarrow$
$append(\#E, \#D, \&R).$
$\}$

以上より，次のようなルールができる．

ルール 3 *segs:cons*

$segs([\&A|\&R], \&S)$
$\rightarrow segs(\&R, \#S1),$
$\langle \#S2, f_{\#Y,\phi(seg([\&A|\&R],\#Y))}, Q_{seg} \cup Q_{ans} \rangle,$
$append(\#S1, \#S2, \&S)$

$Q_{seg} = \{$
$seg(\#K, \#L) \leftarrow append(\#M, \#D, \#K),$
$append(\#C, \#L, \#M),$
$[\#L \neq []].$
$\}$
$Q_{ans} = \{$
$\phi(seg([\&A|\&R], [\&A|\#E])) \leftarrow$
$append(\#E, \#D, \&R).$
$\}$

4.4.4 新述語の生成

4.4.2節と4.4.3節で *segs* アトムを再帰呼出しするルールが生成されるが，その再帰節の中には *setof* 参照が残っている．*setof* 参照は，メタルールから直接得られる等価変換ルールでも計算が可能だが，複数の宣言的記述を並行して計算するため，計算効率が良くない．このため，この *setof* 参照を，参照を使わないアトム列に変換することを考える．

ルール 3の右辺に残る *setof* 参照は，

$\langle \#S2, f_{\#Y,\phi(seg([\&A|\&R],\#Y))}, Q_{seg} \cup Q_{ans} \rangle$
$Q_{ans} = \{$
$\phi(seg([\&A|\&R], [\&A|\#E])) \leftarrow$
$append(\#E, \#D, \&R).$
$\}$

この *setof* 参照から入力 $[\&A|\&R]$ と出力$\#S2$ を取り出し，それらを引数として，新しい述語名 p を持つアトムを考える．新たに生成した p アトムは，上の setof 参照に変換される．これを実現する次のようなメタルールを用意する．

$p([*A| * R], *S)$
$\Leftrightarrow \langle *S2, f_{\%Y,\phi(seg([*A|*R],\%Y))}, Q_{seg} \cup Q_{ans} \rangle,$
$Q_{seg} = \{$
$seg(\#K, \#L) \leftarrow append(\#M, \#D, \#K),$
$append(\#C, \#L, \#M),$
$[\#L \neq []]$
$\}$
$Q_{ans} = \{$
$\phi(seg([\&A|\&R], [\&A|\#E])) \leftarrow$
$append(\#E, \#D, \&R).$
$\}$

このメタルールによって，4.4.3節のメタ節 *c17* を変換すると，setof参照のない *segs* の変換ルールが生成される．

ルール 4 *segs:noset*

$segs([\&A|\&R], \&S)$
$\rightarrow segs(\&R, \#S1),$
$p([\&A|\&R], \#S2),$
$append(\#S1, \#S2, \&S)$

4.4.5 新述語からのルール生成 (1)

新述語 p を変換するルールを生成する．p は第1引数が cons の形で呼び出されることが決まっている．また，cons の形で呼び出された p アトムは4.4.4節のメタルールで変換されるが，それ以上は変換されないので，このメタルールと同じ形の等価変換ルールが得られる．

さらに，第1引数を具体化した形でルール生成する．

c1:$h \leftarrow p([\&A], \&S).$

これは，第1引数が 唯一の要素を持つリストの場合である．

p を展開 → *append* を展開 → 2つの等式制約を解消 → setof の持ち上げ変換を適用 → setof の消去変換を適用 → 1つの等式制約を解消

c8:$h \leftarrow [\&S = [[\&A]]]$.

即ち，次の等価変換ルールが得られる．

ルール 5 *p:single*

$p([\&A], \&S) \rightarrow [\&S = [[\&A]]]$

4.4.6 新述語からのルール生成 (2)

次に，アトム p の第1引数が，2つ以上の要素を持つリストである場合を考える．

c1:$h \leftarrow p([\&A, \&B|\&R], \&S)$.

p を展開 → *append* を展開 → 3つの等式制約を解消 → setof の繰り込み変換を適用 → setof の共通解の持ち上げを適用 → setof の参照関数の書き換えを適用 → p を畳み込むといった変換で，次のメタ節が得られる．

c10:

$h \leftarrow [\&S = [[\&A]|\#S1]],$
$\quad p(\&R, \#S2),$
$\quad distr(\&A, \#S2, \#S1).$

以上により，次の等価変換ルールが得られる．

ルール 6 *p:multi*

$p([\&A, \&B|\&R], \&S)$
$\rightarrow [\&S = [[\&A]|\#S1]],$
$\quad p(\&R, \#S2),$
$\quad distr(\&A, \#S2, \#S1)$

4.5 基本的なルールの用意

setof 参照の変換ルール以外に，以下のような基本的なアトムの計算ルールを準備しておく．これらのアトムの変換ルールの詳細には触れない．

append(X,Y,Z) “リスト X とリスト Y を連結したリストが Z である” という関係を表す述語 *append* を計算するルール．4.3節のメタルールをそのままルールとして書き換えたものを用いる．

equal(X,Y) 等式制約を表す述語 *equal* を変換するルール．X と Y の単一化を行う．

distr(X,Y,Z) “リスト Y の各要素リストの先頭に X を加えたリストは Z である” という意味を持った述語を変換するルール．

これらの変換ルールと，ルール生成で新たに作られるルールとを合わせたものが，部分列問題を解決するためのルール集合となる．

4.6 結果

4.4節で行ったルール生成により，*segs* は次のルール群によって計算可能になる*.

ルール 2 segs:nil

$segs([\,], S) \rightarrow [S = [\,]]$

ルール 4 segs:noset

$segs([A|R], S)$
$\rightarrow segs(R, S1),$
$\quad p([A|R], S2),$
$\quad append(S1, S2, S)$

ルール 5 p:single

$p([A], S) \rightarrow [S = [[A]]]$

ルール 6 p:multi

$p([A, B|R], S)$
$\rightarrow [S = [[A]|S1]],$
$\quad p(R, S2),$
$\quad distr(A, S2, S1)$

append の変換ルール

equal の変換ルール

distr の変換ルール

これらのルールによって，segs アトムを，setof 参照を介さずに計算することができる．

5 ルール生成による *segs2* 問題の解決

5.1 segs2 問題の定義

2.1節で定義したプログラム生成問題において，条件 q を $seg(X,Y)$ 以外の述語にして $P(X)$ を求める例を挙げる．

新たな述語 *seg2* を定義する．$seg2(X,Y)$ は次のような関係を表す．

「Y は，与えられた製数列 X の，必ずしも隣りあわない要素からなる，空でない部分列である．」

ただし，整数列 X の中で先に現われる要素は，整数列 Y でも先に現われるものとする．このとき，与えられた X に対して $seg2(X,Y)$ を満たす Y の集合 S を求めるプログラムを生成する問題を考える．すなわち，

$find\, P$
$\quad such\, that\, P(X) = S = \{Y|seg(X,Y)\}$

である．この問題を，“segs2 問題” と呼ぶ．

---

* 厳密には，これらのルールの左辺に現われる変数を &変数，それ以外の変数を #変数に置き換えたものが，正しい等価変換ルールの記述である．

5.2 setof参照を用いたsegs2問題の表現

$segs2$問題で得られるプログラムが解決すべき問題を，setof参照を含んだ宣言的記述で表現する．整数列Xと$seg2(X,Y)$を満たすYの集合Sとの関係を述語$segs2$で表す．このとき，確定節による$segs2$の定義は次のようになる．

$Q_{segs2} = \{$
$segs2(X,S) \leftarrow \langle S, f_{Y,seg2(X,Y)}, Q_{seg2}\rangle.$
$\}$

ただし，

$Q_{seg2} = \{$
$seg2(X,Y) \leftarrow seg0(X,Y), [Y \neq []].$,
$seg0([],[]) \leftarrow .$,
$seg0([A|X],[A|Y]) \leftarrow seg0(X,Y).$,
$seg0([A|X],Y) \leftarrow seg0(X,Y).$
$\}$

ここで，$\langle S, f_{X,seg(X,S)}, Q_{seg}\rangle$は「$seg(X,S)$を満たす$X$の集合が$S$である」という意味のsetof参照である．

5.3 segs2問題を解くためのメタルール

次に，segs2問題の解となるルールを生成するために，メタルールを用意する．3.4節のsetof参照を変換するメタルールに加えて，segs2アトム，seg2アトム，seg0アトムを変換するメタルールを用意する．

【segs2の宣言的記述から得られるメタルール】

r-4：

$segs2(*X,*S) \Leftrightarrow \langle *S, f_{\%Y,seg2(*X,\%Y)}, Q_{seg2}\rangle$

$Q_{seg2} = \{$
$seg2(\%X,\%Y)$
$\leftarrow seg0(\%X,\%Y), [\%Y \neq []].$,
$seg0([],[]) \leftarrow .$,
$seg0([\%A|\%X],[\%A|\%Y])$
$\leftarrow seg0(\%X,\%Y).$,
$seg0([\%A|\%X],\%Y) \leftarrow seg0(\%X,\%Y).\}$

【seg2の宣言的記述から得られるメタルール】

r-5：

$seg2([*X],*Y) \Leftrightarrow [*Y = [*X]]$

【seg0の宣言的記述から得られるメタルール】

r-6：

$seg0([],*Y) \Leftrightarrow [*Y = []]$
$seg0([*A|*X],*Y)$
$\Leftrightarrow [*Y = [*A|\%Z]], seg0(X,\%Z)$
$\Leftrightarrow seg0(*X,*Y).$

5.4 ルール生成の結果

以上のメタルールを用いて，segs2問題を解く．

1. メタ節
   $h \leftarrow segs2([\&A],\&S).$
   をメタ変換すると，
   $h \leftarrow [\&S = [[\&A]]].$
   が得られる．ここからルール
   $segs2([A],S)$
   $\rightarrow [S = [[A]].$
   が生成される．
2. メタ節
   $h \leftarrow segs2([\&A,\&B|\&R],\&S).$
   を変換すると，
   $h \leftarrow [\&S = [[\&A]|\#L]],$
   $segs2([\&B|\&R],\#S),$
   $segs2([\&B|\&R],\#U),$
   $distr(\&A,\#U,\#T),$
   $append(\#S,\#T,\#L).$
   が得られる．ここからルール
   $segs2([A,B|R],S)$
   $\rightarrow [S = [[A]|L]],$
   $segs2([B|R],V),$
   $segs2([B|R],U),$
   $distr(A,U,T),$
   $append(V,T,L)$
   が生成される．ここで$distr(A,U,T)$は，リストUの各要素リストの先頭にAを追加したリストがTであることを表す．また，$append(S,T,L)$は，SとTを連結したリストがLであることを表す．

これらのルールと$distr$および$append$を変換するルールが，segs2問題の解となる．

5.5 生成されたルールの検証

setofを使わずに$seg2$を求めるルールには，次のようなものが考えられる．

$segs2([],S)$
$\rightarrow [S = []]$
$segs2([A|R],S)$
$\rightarrow segs2(R,U),$
$distr(A,U,T),$
$append(U,T,S)$

このルールは，5.4節で生成されたルールに類似している．しかし，このルールを *segs2* の自然な定義から考え出すのは容易ではない．

ルール生成を用いることにより，segs2 問題の自然な定義からより効率的なルールを生成できることがわかる．

5.6 setof 参照を用いたルール生成の意義

segs 問題と segs2 問題は，集合の要素を決める条件が変わった以外は同一の問題であるが，生成されるルール集合の形は大きく異なる．setof 参照を用いて問題記述に集合表現を導入すると，2つのプログラム生成問題は集合の条件を書き換えるだけで定式化できる．すなわち，問題の変更点がそのまま問題記述の変更点に対応している．ルールはメタ問題記述から自動的に生成できるので，実際にユーザが書き換えるのは条件の部分だけでよい．これに対し，集合表現が存在しない場合は集合表現を含む問題を記述できないため，2つの異なるプログラムを直感で作らなくてはならない．問題の変更点は条件の部分のみであるにも関わらず，segs 問題の解のプログラムから条件にあたる部分を抽出して segs2 問題の解のプログラムに変換することはできない．

問題記述に setof 参照を導入すると，集合表現を含む問題を自然に記述できるばかりではなく，問題の変更点を問題記述に自然に反映することでプログラム生成のコストを軽減することができる．

6 まとめ

本論文では，部分列問題を表す述語 *segs* を setof 参照を用いて表現し，setof 参照や述語 *append*, *distr* の基本的な等価変換を行うメタルールから，setof 参照を用いずに *segs* を計算するルールを生成した．

本論文ではメタルールの概念を採用し，setof 参照の定義からメタルールを作り，メタ節の変換を行うことによって，ルール生成を行った．この方法によって，自然な問題記述から，任意の条件を満たす集合を求める，より効率的なプログラムを自動的に生成することができる．

参考文献

[1] 辻武士，赤間清，宮本衛市：条件を満たす項の集合の表現と計算，電子情報通信学会技術研究報告，SS99-5(1999).

[2] 赤間清，岡田浩一，繁田良則，宮本衛市：参照を含む宣言的記述の定義と負参照の等価変換の正当性，情報処理学会，プログラミング研究会 (1999).

[3] 赤間清，繁田良則，宮本衛市：論理プログラムの等価変換による問題解決の枠組，人工知能学会誌，Vol.12, No.2, pp.90-99 (1997).

[4] 岩崎英哉，胡振江，武市正人：変換部品の組合せによるプログラムの最適化，日本ソフトウェア学会第 15 回大会，D6-3 (1998).

[5] Takashi Yokomori : A Note on the Set Abstractionin Logic Programing Language , Proceedings of The InternationalConference on Fifth Generation Computer Systems (1984).