数理モデル化と問題解決 27－11
（1999．11．26）

# 条件部と実行部を持つ等価変換ルールの自動生成

小池英勝[†] 赤間清[††] 宮本衛市[†]

仕様から効率的なプログラムを自動合成することを目的として，我々は，問題を宣言的に記述した確定節の集合から，多数の正当な等価変換ルールの集合を生成する方法[4]と，そのための基礎理論を提案している[5]．これまでに提案したルール生成法は，対象が条件部と実行部を持たない等価変換ルールであった．本論文では，条件部と実行部を持った等価変換ルールを自動生成する方法を提案する．条件部と実行部を持つ等価変換ルールの自動生成は，自動合成可能なプログラムの効率を大きく改善する．

# Automatic Generation of Equivalent Transformation Rules That Have Condition Atoms and Execution Atoms

HIDEKATSU KOIKE,[†] KIYOSHI AKAMA[††] and EIICHI MIYAMOTO[†]

In order to generate correct and efficient programs from specifications automatically, we have proposed a method of generating many correct equivalent transformation (ET) rules from declarative descriptions (sets of definite clauses)[4] and theoretical foundation for the method[5]. We can generate only ET rules that have neither condition atom nor execution atom by using the method. This paper proposes a method of generating ET rules that have condition atoms and execution atoms. This new method much improves efficiencies of programs that can be generated automatically.

## 1. はじめに

人にとって分かりやすい記述から，正当で効率的なプログラムを自動合成できれば，問題解決に大きく役立つ．本研究の究極的な目標は，自然言語から計算機で実行可能なプログラムを自動合成することである．しかし，このことは，にわかに達成することは難しいと考えられる．そこで我々は，まず始めに，解くべき問題を人が確定節の集合で宣言的に記述して，そこから計算機で実行可能なプログラムを自動合成する研究に取り組むことにした．本研究ではプログラム合成を，等価変換を用いた問題解決の枠組を用いて行う．この枠組では，節の集合をプログラムではなく，問題の関係だけが記述された，手続き的意味を全く含まない仕様と捉える．そして，節の集合から，その宣言的意味[1]を保存したまま変換する等価変換ルールの集合を作る．等価変換ルールの集合は問題を解くための手続きを記述したものであり，これをプログラムとする．よって，「プログラム合成＝等価変換ルールの集合の生成」である．本論文では，これまでに提案した等価変換ルールの生成法を拡張し，条件部と実行部を持った等価変換ルールを自動生成する方法を提案する．本方法でプログラム生成を行えば，人は，節やアトムの順序を全く気にしないで，問題を記述できる．しかも，生成されるプログラムは，高度に効率化されている．また，本方法で扱う問題の対象は節で記述できる全ての問題である．

## 2. 問題記述

本論文では，解くべき問題を表した確定節の集合を問題記述と呼ぶ．問題記述の例として問題記述 $fact$ を以下に示す．

**問題記述** $fact$

$$fact(X,Y) \leftarrow GT(X,0), sub1(X,X1), fact(X1,X2), mult(X,X2,Y).$$
$$fact(0,1) \leftarrow .$$
$$ans(X) \leftarrow fact(3,X).$$

問題記述 $fact$ は，階乗の計算の問題を記述している．

† 北海道大学大学院 工学研究科 システム情報工学専攻
Division of System and Information Engineering, Hokkaido University
†† 北海道大学 情報メディア教育研究総合センター
Center of Information and Multimedia Studies, Hokkaido University

ans節は，質問を表す節である．述語$GT$は，大小関係を表す述語である．$sub1$は第1引数から1引いたものが第2引数であるということを表す述語である．$mult$は，積の計算を表す述語である．計算中に等価変換ルールで書き換えられるのは質問の節だけである．

## 3. 等価変換ルール

本論文では，問題記述の節に等価変換ルールを適用して変換することを，変換，又は，計算と呼ぶことがある．問題記述$fact$を解くための条件部と実行部を持った等価変換ルール$r1, r2$を以下に示す．

$\langle r1\rangle$:
$fact(X,Y), \{int(X), X > 0\}$
$\quad \rightarrow \{E := X - 1\}, fact(E,F), mult(X,F,Y).$
$\langle r2\rangle$:
$fact(0,Y)$
$\quad \rightarrow \{Y = 1\}.$

$r1$は，factアトムが第1引数$X$が0より大きい整数のとき適用可能である．適用されるとまず始めに$X-1$の演算を行い，結果を変数$E$に代入する．そして，節にあった元の$fact$アトムを展開部のアトム$fact(E,F), mult(X,F,Y)$に書き換える．$r2$は，$fact$アトムが第1引数$X$に0を持つとき適用可能で，適用されると第2引数$Y$に1を代入する．

## 4. メタ表現とメタルールの導入

### 4.1 メタ表現の導入

節や問題記述を抽象化する表現を導入する．本論文では，この表現をメタ表現と呼ぶ．メタ表現は，&変数，#変数，メタ項，メタアトム，メタ節，メタ問題記述から成る．&変数は，先頭に&のついた変数で記述し，任意の項を表す．#変数は，先頭に#のついた変数で記述し，任意の変数を表す．メタ項は，通常の変数の代わりに&変数や#変数が現れる項である．メタアトムは，引数にメタ項が現れるアトムである．メタ節は，ダミーヘッドhとメタアトムから構成され，以下のような形をしている．

$h \leftarrow fact(\&X, \&Y). \ldots\ldots\ldots (mcl1)$

節の$h$は，任意のヘッドアトムを表す．このメタ節は，ボディに2つの任意の引数を持つfactアトムが現れる全ての節を表す．メタ問題記述は，メタ節の集合である（空集合も許す）．メタ問題記述は通常の問題記述を抽象化したものである．メタ節$mcl1$から成るメタ問題記述は，例えば，本論文に現れる問題記述$fact$を表す．

### 4.2 メタルール

通常の問題記述を等価変換するルールを等価変換ルールと呼ぶのに対して，メタ問題記述を等価変換するルールをメタルールと呼ぶ．メタルールの例$a1$を以下に示す．

$\langle a1\rangle$:
$fact(*A, *B)$
$\quad \rightarrow equal(*A, \%C), equal(*B, \%D),$
$\quad\quad GT(\%C, 0), sub1(\%C, \%E),$
$\quad\quad fact(\%E, \%F), mult(\%C, \%F, \%D);$
$\quad \rightarrow equal(*A, 0), equal(*B, 1).$

等価変換ルールと似た形をしているが，変換対象がメタ問題記述（メタ節の集合）であること，現われる変数が＊変数と％変数であること，ヘッドに複数のアトムを許すことが異なる．＊変数とは先頭に＊のついた変数で，任意のメタ項が代入できる．％変数とは，先頭に％のついた変数で#変数が代入できる．本論文では，メタ問題記述のメタ節にメタルールを適用して変換することを，メタ変換，又は，メタ計算と呼ぶ．

### 4.3 制約付き変数の導入

条件部や，実行部の付いた等価変換ルールを生成するために制約付き変数を導入する．例えば，$fact(3,X)$を$fact(\&X : \{int(\&X)\}, \&Y)$とすることで，第1引数が任意の整数である$fact$という抽象化が行える．制約は&変数にのみ付く．メタ問題記述中の&変数に制約を付け加えることを，本論文では特殊化と呼ぶことにする．

### 4.4 要求駆動のルール生成

等価変換ルールを生成する時，どのようなルールを作ればよいかは，質問を表す節($ans$節)から決めることができる．そこで，要求駆動のルール生成システムを考える．このシステムは，適用可能な等価変換ルールが存在するときは，その等価変換ルールを使って計算を行う．計算が終了していないのに，適用可能な等価変換ルールが存在しない時は，計算が止まった時点の質問の節を参照して，その節を変換できるルールを新しく生成する．計算とルール生成を，解が得られるまで交互に繰り返し，計算が終了したとする．計算が終了すると，必要な等価変換ルールが全て生成されたことになる．生成された等価変換ルールは，他の質問にも適用可能なルールである☆．

☆ 他の質問にどの程度適用できるかは，生成されるルールによって異なる．

### 4.5 実際の質問の節の情報

質問の節からメタ節を作る時，実際の質問の節に現れる具体的な引数を，そのまま制約に加える．この引数を中心値と呼ぶことにする．中心値には，

- 具体的な数値を使った効率化
- 生成された等価変換ルールが，要求された節（計算が止まった時点の$ans$節）を必ず変換することを保証する

という役割がある．例として，

$ans(X) \leftarrow fact(3, X)$.

からは，

$h \leftarrow fact(\&X : \{3\}, \&Y)$.

のようなメタ節を作る．3が中心値で，&Xが3にまで特殊化化可能であることを示す．本論文では，中心値を制約の一番先頭に記述することにする．

### 4.6 あらかじめ用意するメタルール

あらかじめ用意するメタルールとは，等式制約や公式など，異なる問題に共通に用いることのできるメタルールである．以下に具体例を示す．

$\langle m1 \rangle$:

$equal(*X, *Y)$,
$\{isSVar(*Y), notInclude(*Y, *X)\}$
$\rightarrow \{*X = *Y\}$.

$m1$は，等式制約を扱うルールで，メタ節のボディにある$equal$メタアトムについて，第2引数$*Y$が#変数で，かつ，第1引数$*X$が，その#変数を含まないならば，適用可能で，適用されると$equal$メタアトムは第1引数と第2引数の単一化を実行する．そして，実行が成功した場合は，節から$equal$アトムを消去し，失敗した場合は，適用された節を消去する．$isSVar$は，与えられた引数が#変数かどうかを調べる述語である．$notInclude$は変数を含むかどうかを調べる述語である．

$\langle m2 \rangle$:

$equal(*X, *Y)$,
$\{*X == *Y\}$
$\rightarrow \{true\}$.

$\langle m3 \rangle$:

$equal(*X, *Y)$,
$\{getInfo(*X, [*C| * L])$,
$int(*Y), int(*C)$,
$*Y! = *C\}$
$\rightarrow \{putInfo(*X, [*C, int(*X)$,
$(*X! = *Y)| * L])$,
$false\}$.

$\langle m4 \rangle$:

$equal(*X, *Y)$,
$\{getInfo(*X, [*C| * L])$,
$int(*Y), int(*C)$,
$*Y == *C\}$
$\rightarrow \{rmInfo(*X), *X = *C\}$.

$\langle m5 \rangle$:

$sub1(*X, *Y)$,
$\{getInfo(*X, [*C| * L])$,
$int(*C), isSVar(*Y)\}$
$\rightarrow \{*C2 := *C - 1$,
$putInfo(*Y, [*C2, int(*Y)$,
$(*Y := *X - 1)| * L])$,
$putInfo(*X, [*C, int(*X)| * L])\}$.

$\langle m6 \rangle$:

$GT(*X, *Y)$,
$\{getInfo(*X, [*C| * L])$,
$not(member(*X > *Y) * L)$,
$int(*C), int(*Y)$,
$*C > *Y\}$
$\rightarrow \{putInfo(*X, [*C, *X > *Y$,
$int(*X)| * L])\}$.

$\langle m7 \rangle$:

$GT(*X, *Y), \{int(X), int(Y)\}$
$\rightarrow \{*X > *Y\}$.

メタルール$m2$は，$*X$と$*Y$が等しい時$equal$を消去する．$m3$は，生成される等価変換ルールの適用条件を狭め，ボディを減らす働きをする．$m5$は「$*X$から，1引いたものが$*Y$である.」という制約を処理するルールである．$m6, m7$は，大小関係を扱う．

## 5. メタ問題記述の変換

以降は制約付き変数を節の外側に記述することがある．例として

$h \rightarrow atom1(\&X : \{1, (\&X > 0)\}, \&Y : \{2\})$,
$atom2(\&X : \{1, (\&X > 0)\})$.

と

$h \rightarrow atom1(\&X, \&Y), atom2(\&X)$.
$\&X : \{1, (\&X > 0)\}$
$\&Y : \{2\}$

は，全く同じメタ問題記述を表すことにする．

### 5.1 メタ計算

始めに，本節の$step1$にあるようなメタ問題記述を作ったとする．メタ計算では，そのメタ問題記述をメタルール$a1$，$m1 \sim m7$で変換していく．

変数に付く制約は，ステップを越えて反映される．

具体例で説明する．本節に現れる，$step3$ で $\&X$ には，$\&X : [3, \&X! = 0]$ という制約が付加されているが，この時 $step1$ の $\&X$ の制約も $\&X : [3, \&X! = 0]$ に変わる．以降のステップでも同様のことが起こる．

以下に，$r1$ を生成するためのメタ計算の過程を示す．

step1

$h \leftarrow fact(\&X, \&Y).$

$\&X : [3]$

step2

$h \leftarrow equal(\&X, \#B), equal(\&Y, \#D),$
$GT(\#B, 0), sub1(\#B, \#E),$
$fact(\#E, \#F), mult(\#B, \#F, \#D).$

$h \leftarrow equal(\&X, 0), equal(\&Y, 1).$

$\&X : [3]$

$(a1)$ より

step3

$h \leftarrow equal(\&X, \#B),$
$equal(\&Y, \#D), GT(\#B, 0), sub1(\#B, \#E),$
$fact(\#E, \#F), mult(\#B, \#F, \#D).$

$\&X : [3, \&X! = 0]$

$(m3)$ より

step4

$h \leftarrow equal(\&Y, \#D), GT(\&X, 0),$
$sub1(\&X, \#E),$
$fact(\#E, \#F),$
$mult(\&X, \#F, \#D).$

$\&X : [3, \&X! = 0]$

$(m1)$ より

step5

$h \leftarrow GT(\&X, 0),$
$sub1(\&X, \#E),$
$fact(\#E, \#F),$
$mult(\&X, \#F, \&Y).$

$\&X : [3, \&X! = 0]$

$(m1)$ より

step6

$h \leftarrow sub1(\&X, \#E),$
$fact(\#E, \#F),$
$mult(\&X, \#F, \&Y).$

$\&X : [3, int(\&X), \&X > 0]$

$(m6)$ より

step7

$h \leftarrow fact(\&E, \#F),$
$mult(\&X, \#F, \&Y).$

$\&X : [3, int(\&X), \&X > 0]$

$\&E : [2, int(\&E), \&E := \&X - 1]$

$(m5)$ より

### 5.2 変換列からの等価変換ルール生成

$step1$ から $step7$ までの変換は，全て等価変換である．よって $step1$ のメタ問題記述が表す問題記述を $step7$ のメタ問題記述が表す問題記述に書き換える変換も等価変換である．このことから新しい等価変換ルールを作ることが出来る．等価変換ルールの生成は，変換列の最初と最後のメタ問題記述から，ヘッド，条件部，実行部，展開部を作ることで行う．ヘッドは，$step1$ の節のボディを用いる．条件部は $step1$ の節のボディアトムに現れた変数の制約を使う．この例では，$\&X$ に制約 $int(X)$ と $\&X > 0$ が付いているので，この制約を条件部に入れる．実行部は，制約の中の演算を表すものから作る．この例では，$\&E$ に制約 $\&E := \&X - 1$ があるので，この演算を実行部に入れる．展開部は，$step7$ のメタ節のボディアトムから作る．最後に変数を通常の変数にする．この様にして $r1$ が作られる．

## 6. ま と め

本論文では，問題記述から，条件部と実行部を持った等価変換ルールを生成することで，効率的なプログラムを自動生成する方法を提案した．本論文で示したルール生成を自動的に行うシステムは既に試作されている．現在は，様々な例題をシステムに与え，多くの異なる性質の問題の効率化が行えるように，あらかじめ用意しておくメタルールを充実させることが課題である．また，与える問題記述を一階述語論理表現まで拡張する予定である．

## 参 考 文 献


1) 赤間清, 繁田良則, 宮本衛市：論理的問題の等価変換による解法（1），人工知能学会誌，Vol.13, No.6, pp.928-935 (1998).
2) 赤間清, 繁田良則, 宮本衛市：論理的問題の等価変換による解法（2），人工知能学会誌，Vol.13, No.6, pp.936-943 (1998).
3) 淵　一博, 古川　康一, 溝口　文雄：プログラム変換, 共立出版 (1987)
4) 小池英勝, 赤間清, 宮本衛市：仕様からの等価変換ルールの生成法, 電子情報通信学会技術研究報告 KBSE98-8, pp.33-40 (1998)
5) 小池英勝, 赤間清, 宮本衛市：等価変換ルールの生成方法の理論的基礎, 情報処理学会研究報告 98-ICS-144, pp.13-18 (1998)
6) 小池英勝, 赤間清, 宮本衛市：等価変換ルールの探索に基づくプログラム合成, 電子情報通信学会技術研究報告 SS99-20, pp.41-48 (1999)