TM-0424

項書換えに基づく帰納的定理証明

大須賀昭彦，坂井　公

December, 1987

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

03) 456-5191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

項書換えに基づく帰納的定理証明

An Inductive Theorem Proving Method based on Term Rewriting

大須賀昭彦，　坂井公
Akihiko OHSUGA, Ko SAKAI

（財）新世代コンピュータ技術開発機構
Institute for New Generation Computer Technology

Abstract A method that proves inductive theorems in equational theory is described. Huet and Hullot have developed a method called inductionless induction to prove inductive theorems without explicit induction. Our method is an extension of their method and can avoid failure and infinite execution whereas their method sometimes fails or loops infinitely.

あらまし 等号論理における帰納的定理を証明する手続きについて述べる．通常この種の定理証明は帰納法を必要とするが，我々の方法はHuet,Hullot らによって提案された潜在帰納法(inductionless induction) に基づき，帰納法を用いずに証明を行う．また，従来のこの種の手続きは (1)証明成功 (2)証明失敗 (3)無限実行の３種の結果をもったが，(2)(3)の回避について考察する．

１．はじめに

HusserやGoguenは，等号論理における帰納的定理の成立が，その公理系に問題の定理を新たな公理として追加して得られる拡張公理系の無矛盾性と等価なこと，さらには，Knuth&Bendixの完備化手続き[Knuth 70]がその無矛盾性の判定に利用できることを示した[Husser 80] [Goguen 80]．彼らの方法が一種類の矛盾判定(true=false)しかもたなかったのに対し，HuetとHullotは関数記号を構成子と演算子に分割することによって，矛盾の検出を容易に行う方法を提案した[Huet 82]．通常，帰納法を必要とするこの種の定理証明に対し，直接に帰納法を使うことなく証明を実現するこの方法は，潜在帰納法(inductionless induction) と呼ばれ，最近さかんにその応用が研究されている．この手続きは，等号論理の公理系Ｅに証明すべき等式を加え新しい公理系Ｅ' をつくったとき，ＥとＥ' が等しい始モデルを持てば加えた等式はＥ上の定理であるという考え方に基づいている．今回，項書換えシステム生成系Metis 上に潜在帰納法を実現したので，その証明手続きについて報告する．

まず，２節で用語と記法を定義し，３節では項書換えシステム生成系Metis の特徴について述べる．４節で潜在帰納法の定義とその拡張を行い，アルゴリズムについて５節で解説する．最後に，６節でいくつかの実験例を報告する．

２．準備

本節では，本稿で必要な記法と用語について定義する．TRS に関する基礎的な概念については既知とする[Huet 80][坂井 87]．

Ｆを関数記号(function symbol) の集合，Ｖを変数(variable)の集合，T(F,V)をＦ，Ｖからつくられる項(term)の集合，T(F)をＦのみからつくられる項，つまり変数を含まない基礎項(ground term) の集合とする．関数記号は固定アリティをもつものとし，定数は０引数の関数と考える．

項ｔが部分項としてｓをもつことをt[s]と表し，t[s'] でｔにおける sの出現をs'で置換え

たものを表す．ここでは特に，sは変数でないものとする．t[s1,...,sn]等も同様に定義する．

【定義 2.1】T(F,V)上の２項関係 l→r の有限集合を**項書換えシステム**(Term Rewriting System; 以降TRS と記す) といい，l→r を**書換え規則**(rewrite rule)という．□

【定義 2.2】項 t[s]は，s=θ(l) なる代入(substitution)θが存在するとき，書換え規則 l→r によって，t[θ(r)]に**簡約**(reduction)される．これを，t[s]⇒t[θ(r)]と表す．また，⇒の反射推移閉包(reflexive transitive closure)を↠で表わす．□

【定義 2.3】項t に関して，t⇒u なるu が存在するならば，tを**可約項**(reducible term)といい，そうでなければ，tを**既約項**(irreducible term)という．t↠u かつ u が既約項であるとき，u をｔの既約形といい，t↓で表す．□

【定義 2.4】TRS Rにおいて，t0⇒t1⇒....なる無限の簡約が存在しないとき，Rは**停止性**(termination) をもつという．□

【定義 2.5】TRS Rにおいて，任意の項ｔに対し，t↠u，t↠v ならば u↠w，v↠w なる項ｗが必ず存在するとき，Rは**合流性**(confluency)を持つという．□

【定義 2.6】停止性と合流性を同時に持つTRS を**完備な**(complete)TRS と呼び，任意の TRSに論理的に等価な変型を施して，完備な TRSを得る操作を**完備化**(completion)という．完備な TRS においては，ｔの既約形ｔ↓が一意に定まる．これをｔの**正規形**(normal form) という．□

【定義 2.7】２つの書換え規則g[s]→d，l→r が与えられたとき，θ(s)=θ(l) なる代入θが存在するならば，θ(d)=θ(g[r])のことを，g[s]→d に l→r を**重ねた要対**(critical pair) という．また，これを求める手続きを**要対法**(critical pair method)，項θ(g[s])をこれら２規則間の**重像**(superposition) という．□

【定義 2.8】EをT(F,V)上の等式の集合とし，Eによる合同関係を～とするとき，T(F)/～をT(F,V)のEによる**始モデル**(initial model) という．□

【定義 2.9】EをT(F,V)上の等式の集合とする．T(F,V)上の等式m=n がEの**帰納的定理**であるとは，m=n がEによる始モデルで常に成立することをいう．□

定義2.9 は，m=nに対して任意の基礎代入(ground substitution) θを行ったとき，θ(m) ～θ(n) となることを意味している．

【定義2.10】TRS Rにおいて，任意の基礎項がt0⇒t1⇒....のように無限に簡約されないとき，Rは**基礎停止性**(ground termination)をもつという．□

【定義2.11】TRS Rにおいて，任意の基礎項ｔに対し t↠u，t↠v ならば u↠w，v↠w なる項ｗが必ず存在するとき，Rは**基礎合流性**(ground confluency) を持つという．□

【定義2.12】T(F)上で，停止性と合流性を同時に持つTRS を**基礎完備な**(ground complete)TRSと呼ぶ．□

３．項書換えシステム生成系Metis[Ohsuga 86]

次に，本稿の実験環境を提供している実験システムMetis の特徴について述べる．

Metisは，TRSに関する種々の技術研究を目的とした，汎用のTRS 生成系であり，現在DEC-2060とPSI （各々DEC10-Prologと ESPで記述）に実装されている．図3.1.にMetis のシステム構成を示す．データベースには，等式，書換え規則の他に，関数記号に関する情報及び項の順序付に関するデータが蓄えられる．処理系には，等式を書換え規則化する際の停止性に関する推論を行う停止性推論部，TRSの完備化を行う完備化手続き部，いくつかの戦略によって等式の定理証明を行う等式証明部，TRSによる項の簡約を実行する簡約部，入出力等を行うユーティリテ

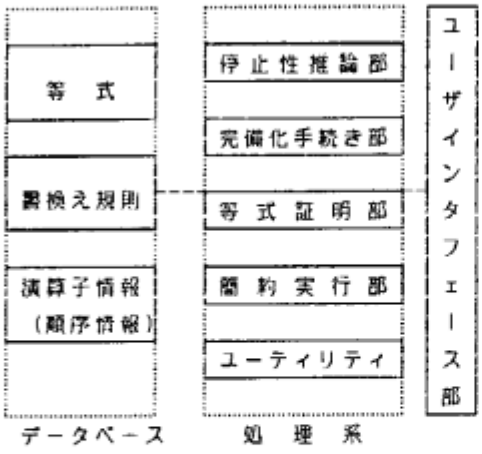

図-3.1. Metis システム構成

ィ部等をもつ．Metis 上の処理は，ユーザインタフェース部を通じてすべて会話的に行われる．以下にMetis の主な機能について述べる．

3.1.停止性の保証

簡約の停止性保証問題は一般に決定不能である．Metis ではこれに対し，複数の単純化順序による停止性の保証を試みる．つまり，任意の書換え規則 l→r に対して，l＞r(＞は適当な単純化順序) となることを示し，停止性を保証する．このとき，関数記号間の順序関係が重要かつ複雑となるが，これに関しては，システムが書換え規則の構造から自動的に推論するので，ユーザの判断は必要とされない．以下に単純化順序の定義を述べる．

【定義 3.1】以下の性質をもつ順序＞を**単純化順序** (simplification ordering) [Dershowitz 82] という．

(1) c[s] ≥ s

(2) s≥t ならば c[s] ≥c[t]

(3) s≥t ならば $\theta(s) \geq \theta(t)$ □

具体的な単純化順序としては，①辞書式部分項順序(lexicographic subterm ordering) [Sakai 84a]，②勝抜順序[Sakai 84b]，③多重集合順序(multiset ordering)[Dershowitz 84]等が用意されており，これらは関数記号毎に設定される．特に①②は，後述の強単純化順序にもなっている．以下で，本稿の実験で使われる辞書式部分項順序について定義する．

【定義 3.2】関数記号間に適当な順序＜が導入されているとき，次の順序≪を**辞書式部分項順序**という．

(1) 変数v は，任意の項t に対して t≪v となることはない

(2) (非変数) 項t=g(b1, ...,bn)と項s に対し s≪t となるのは，次のいずれかの場合，かつ，それらに限られる

(2-1)あるbiに対し s≪biとなる

(2-2)s=f(a1, ...an) で，各aiに対しai≪t であり，かつ，次のいずれかが成立する

(2-2-1)f＜gである

(2-2-2)f=g，かつ，あるｉがあり，

a1=b1,...,ai-1=bi-1, ai≪biとなる□

各関数記号は，固定アリティをもつことが仮定されているので，上記(2-2-2) においては，an=bn,...,ai+1=bi+1, ai≪biなる比較も可能である．両比較方法を併用する場合，定義3.2 の順序を特に正順辞書式部分項順序，この方法によるものを逆順辞書式部分項順序と呼ぶことにする．

【例 3.1】項(X･Y)･Z とX･(Y･Z) は，正順辞書式部分項順序によっては，(X･Y)･Z ≫ X･(Y･Z) の向に，逆順辞書式部分項順序によっては，(X･Y)･Z ≪ X･(Y･Z)の向に順序付される．□

3.2.完備化手続き

TRSの完備化機能としては，①Knuth&Bendix の完備化手続きの他に，その拡張として①に結合・交換(associative commutative;AC)単化(unification) を組込んだ②AC完備化手続き[Peterson 81]，向付自由書換え規則や等式の否定を扱う③Ｓ戦略[Hsiang 87] 等が実装されている．

3.3.定理証明機能

等式証明のための機能としては，通常行われる①簡約による証明の他に，Ｓ戦略を利用した②反駁型定理証明，本稿で述べる潜在帰納法による③帰納的定理証明機能等をもつ．①における簡約の戦略は，常に最外最左戦略がとられる．

4. Metisにおける潜在帰納法

以下に，潜在帰納法について述べる．等号論理Eに証明すべき等式を加えた論理E'をつくり，両者の始モデルを比較するというのが，潜在帰納法の考え方であった．E，E'に対応する完備なTRS R，R'において始モデルが一致するとは，Rの基礎項上の既約形がR'でも既約形であることを意味する．そこで，R上の帰納的定理は次の手順で証明可能となる．

(1) Rを完備なTRS,Eを証明すべき等式の集合とする．
(2) R+Eを完備化して，R'を得る．
(3) 得られた完備なTRS R'が，元のRにおける基礎項上の既約形を簡約しないことを確認する．

これが潜在帰納法といわれる定理証明手法の概要である[Huet 82][Sakai 87]．以下にMetisにおける潜在帰納法について述べる．

4.1.定義原理[Huet 82][Sakai 87]

上記(3)の判定は一般に容易ではなく，現に，いくつかの判定方法が提案されているが，我々は，実現の効率を考慮して，定義原理に基づく判定を採用した．以下に，定義原理による，(3)の判定について説明する．

Fは演算子(defined symbol)Dと構成子(constructor) Cに分割されているものとし，T(C,V),T(C) をT(F,V),T(F) 等と同様に定義する．また，Cには少くとも2つの構成子が含まれていることを仮定する．

【定義 4.1】完備なTRS Rが定義原理(principle of definition)をみたすとは，任意の基礎項 t∈T(F)に対し，tがRにおける既約形となるのが t∈T(C)のとき，かつこのときに限られることをいう．□

この原理をみたす完備なTRSに関しては，次の定理により(3)の判定が可能となる．

【定理 4.1】新たな書換え規則がRの既約な項を簡約しない
<=> 書換え規則の左辺 lが l∉T(C,V).
□

また，完備なTRSが定義原理をみたすことを確認する十分条件として，次が知られている．

【定理 4.2】完備なTRS Rにおいて，任意の書換え規則の左辺が d∈Dを含み，かつ任意の d∈D，ci∈C(1≦i≦n)について，項d(c1,...,cn)が可約
=>Rは定義原理をみたす．□

4.2.等式に対する向付失敗の回避

従来の潜在帰納法では，新たな等式に向付ができない場合R'の停止性が保証できなくなり証明は失敗に終わっていた．我々はこの問題に対し，S戦略における向付自由書換え規則の考え方を導入した．これは，単純化順序によって保証されるT(F,V)上の停止性を，強単純化順序によってT(F)上だけに制限して保証するというものである．以下に強単純化順序の定義を述べる．

【定義 4.2】以下の性質をもつ順序>を強単純化順序(strong simplification ordering)という．
(1) c[s] ≥ s
(2) s≥t ならば c[s] ≥ c[t]
(3) s≥t ならば θ(s) ≥θ(t)
(4) >はT(F)上で全順序となる□

(1)-(3)は通常の単純化順序の定義であり，強単純化順序が，T(f,V)上で，部分項の置換えや代入に対し安定な整礎順序であることは，定義より明らかである．

【定理 4.3】関数記号間に全順序が導入されているならば，辞書式部分項順序，勝抜順序は強単純化順序である．□

これに基き，潜在帰納法では，等式の書換え規則への変換を次のように行う．>を適当な強単純化順序，l=rを向付したい等式とする．
(1) l > r (r> l)ならば，書換え規則として
l →r (r→l)を得る．
(2) さもなくば，両方向に適用可能な向付自由規則 l↔r を得る．

これに伴い，要対法，簡約等の定義を以下のように拡張する．また新たに，狭化の定義を導入する．ここでは，向付自由書換え規則の導入により，従来通りに向付のされている書換え規則を，特に向付書換え規則と呼ぶことにする．

【定義 4.3】項g[s]は，次の代入θが存在するとき，l→r によって，g[θ(r)]へ**拡張簡約**(extended reduction)される．
(1) s=θ(l)
(2) θ(l) ＞ θ(r) □

ここで，l→r が向付書換え規則 l→r である場合，(2)は常に成立する．

【定義 4.4】２つの書換え規則g[s]→d，l→r が与えられ，次の代入θが存在するとき，θ(d)=θ(g[r])のことを，g[s]→d に l→r を重ねた**拡張要対**(extended critical pair)という．
(1) θ(s)=θ(l)
(2) θ(l) ≦ θ(r)
(3) θ(g[s])≦ θ(d)
拡張要対を求める手続きを**拡張要対法**(extended critical pair method)といい，θ(g[s])をこれら２規則間の**拡張重像**(extended superposition)という．□

ここで，l→r なら，(2)は常に成立し，g[s]→d なら，(3)は常に成立する．

【例 4.1】項a+b と書換え規則 X+Y→Y+X に対し，+を正順辞書式部分項順序の関数，b<aが関数記号間の順序で成立するものとする．このとき，a+b＞b+a が成立つので，a+bは，X+Y→Y+X によってb+a に拡張簡約される．□

【例 4.2】書換え規則 s(V+W)→ s(W)+Vと X+Y→Y+X に対し，s,+を正順辞書式部分項順序の関数とする．このとき，θ(V+W)=θ(X+Y)なるθ={V←X, W←Y}が存在し，また，いかなる関数記号間の順序を仮定してもθ(X+Y) ≦ θ(Y+X), θ(s(V+W))≦ θ(s(W)+V)) であるため，拡張要対s(Y)+X=s(Y+X) が得られる．このときの拡張重像はs(X+Y)である．□

【定義 4.5】項g[s]は，次の代入θが存在するとき，l→r によって，θ(g[r])へ**拡張狭化**(extended narrowing)される．
(1) θ(s)=θ(l)
(2) θ(l) ≦ θ(r) □

これらの拡張が潜在帰納法において有効となるいのは，潜在帰納法においては，Rが(T(F,V)上）完備でなくても，(T(F)上）基礎完備であれば十分であるという事実による．基礎完備なRは，次の定理により得られる．

【定理 4.4】TRS Rが基礎停止性をもち，かつ，互いの書換え規則間に，発散する拡張要対をもたないならば，Rは基礎完備である．□

4.3.潜在帰納法の効率化

一般に，拡張要対法は，通常の要対法に比べより多くの拡張要対を生成する．これは，直観的にいって，拡張要対法がパラモジュレーションに近い操作であることに起因する．そこで，要対の爆発的増大を防ぐための工夫が必要となる．

【定義 4.6】等式l=r は，次の代入θi(1 ≦ i ≦n)が存在するとき，等式gi=di(gi→di) によって包含(subsume) される．
(1) l=c[θ1(g1),...,θn(gn)]
(2) r=c[θ1(d1),...,θn(dn)]□

他の等式に包含されるような等式が，等式集合E上で，論理的に冗長であることは明らかである．これを取り除く戦略は，向付自由規則を扱う際に有効である．

【例 4.3】等式集合E={s(X+Y)=s(Y+X), V+W=W+V}を考える．s(X+Y)=s(Y+X) は，V+W=W+Vによって包含されているので，これを除いたE'={V+W=W+V}を得ても，Eとは論理的に等価である．□

Fribourgは，潜在帰納法を本来の帰納法と比

較したとき，本来の帰納法が，証明すべき等式に対し，帰納法の適用箇所を１箇所選んで慎重に実行されるのに対し，潜在帰納法は，証明すべき等式に対し，複数の帰納法を同時に適用していることを示した[Fribourg 86]．また，潜在帰納法において，正しい証明を導く適切な帰納項(good induction term)はごく一部であり，適切でない帰納項への潜在帰納法の適用が，証明の無限実行の原因であるとの観察を行った．彼は，これに基づき，適切な帰納項のみを選択して重複を得るという，要対法に対する強い制限を提案した．制限の概要は以下の通りである．Pを（元の）完備なTRS,Eを証明すべき等式の集合，トをEから選択され向付された１つの書換え規則とすると，

(1) 重複は，FにPの書換え規則を重ねるもののみ考慮する．つまり，F同士やFとFから派生した要対との間では，重複を取らない．
(2) 重複は，Fに対し１箇所のみ取られる．つまり，複数の重複が取れる場合，その中の１つが選択される．

我々の方法は，この制限を部分的に採用している．これは，我々の採用した基礎停止性による方法と，彼の制限との整合性検討が必要なためである．彼は，重複選択のために完全重複可能性(complete superposability)の概念を導入したが，我々の方法では，この条件は自然に満足される．

【例 4.4】R={0+X→X, s(X)+Y→s(X+Y)}, E={(A+B)+C=A+(B+C)}, +を逆順辞書式部分項順序による関数とすると，Fとして A+(B+C)→(A+B)+C が得られる．FへRの２番目の規則を重ねると，要対として，
①s(X+(B+C))=(s(X)+B)+C
②(A+s(X))+Y=A+(s(X+Y))
が得られる．ここで，①を選択すると，これは，Fによりs((X+B)+C)=s((X+B)+C)へ簡約され，取除かれるが，②を選択すると，(A+s...s(X))+Y=A+s...s(X+Y) なる要対が次々得られ手続きは停止しなくなる．このように，ここでは，①が適切な帰納項の選択となる．

4.4.無限実行の検出

潜在帰納法は，上記の要対法に対する制限を加えても，無限の要対を生成し続け停止しないことがある．しかし，多くの場合，新たな補題（等式）を追加することによって，そのような無限実行は回避できる．そこで，この状況の検出が重要な問題となってくる． Hermannは，Knuth&Bendix完備化手続きの無限実行を，要対法における書換え規則の重なり方から判定する十分条件を提案した[Hermann 85]．我々は，潜在帰納法の無限実行の検出に，Hermann の交差対(crossed pair)の考え方に類似の再帰対の概念を導入した．

【定義 4.7】２つの書換え規則 g[s]→d, l→r[t]が与えられ，次の代入θ，θ' が存在するとき，θ(d)=θ(g[r[t]]) を再帰対と呼ぶ．
(1) θ(s)=θ(l),
(2) θ'(θ(t))= θ'(l')
(3) θ(g[r[t]]) ≻ θ(d)
ここで，l'はlの変数名を適当に付変えたものを表す．□

再帰対は，定義より明らかなように要対でもある．再帰対が存在し，これを簡約する規則が存在しない場合，次の一般形をした要対が無限に生成される．
θ0(g[r[t]])= θ0(d)
θ1(θ0(g[r[r[t]]]))= θ1(θ0(d))
.....
θn(..θ0(g[r[.r[t].]]).)=θn(..θ0(d)..)
ここで，θn は θn(θn-1(t))=θn(In) なる代入である．

【例 4.5】例4.4 の②の場合について考察する．２つの書換え規則
A+(B+C) → (A+B)+C
s(X)+Y→s(X+Y)
に対し，
(1) θ0(B+C)= θ0(s(X)+Y) なるθ0 が存在
(θ0={B←s(X),C←Y})
(2) θ1(θ0(X+Y))=θ1(s(X1)+Y1)　つまり，θ1(X+Y)= θ1(s(X1)+Y1) なるθ1 が存在
(θ1={X←s(X1), Y←Y1})

これにより，再帰対

(A+s(X))+Y=A+(s(X+Y))

が得られる．実際この２つの書換え規則は無限の要対を生成し続ける．

A+(s(X+Y))=(A+s(X))+Y

A+(s(s(X1+Y1)))=(A+s(s(X1)))+Y1

A+(s(s(s(X2+Y2))))=(A+s(s(s(X2))))+Y2

.....

A+(s(..s(Xn+Yn)..))=(A+s(..s(Xn)..)+Yn

□

Hermann の交差対による方法が，２つの書換え規則g[s]→d，l→r[t]に関して，(1)θ(s)=θ(l)，(2)θ'(t)=sなる条件を検査するのに対し，我々の方法は，かなり緩い条件となっている．これは，交差対の条件がかなり制限されたものであり，実際の証明における無限実行の多くの場合が，この条件にあてはまらないという事実によるものである．現在の再帰対の条件を，実用的な検出能力を低下させずに，無限実行の十分条件に近づけることは，今後の課題である．

向付自由書換え規則を扱うための再帰対の拡張は，以下のように行われる．

【定義 4.8】２つの書換え規則g[s]→d，l→r[t]が与えられ，次の代入θ，θ'が存在するとき，θ(d)=θ(g[r[t]])を拡張再帰対と呼ぶ．

(1) θ(s)=θ(l)

(2) θ(l) $\lesssim$ θ(r[t])

(3) θ(g[s]) $\lesssim$ θ(d)

(4) θ'(θ(t))= θ'(l')

(5) θ(g[r[t]]) $\lesssim$ θ(d)

ここで，l'はlの変数名を適当に付変えたものを表す．□

証明の無限実行を回避するために必要とされる補題は，上記要対の一般形をさらに一般化するような等式である．現在これを自動獲得する方法についても検討中であるが，今のところ，補題の必要性の判断とその獲得はユーザに任されている．

【例 4.6】例4.5 で必要とされる補題は，X+s(Y)=s(X+Y) である．これを書換え規則X+s(Y)→s(X+Y)として，例4.5 の最初の要対を簡約すると，s(A+X+Y)=s(A+X+Y)が得られ，無限の要対生成を防げる．

５．アルゴリズム

以下に，Hetisにおける潜在帰納法のアルゴリズムを与える．以降，拡張要対，拡張簡約，拡張狭化，拡張再帰対を単に要対，簡約，狭化，再帰対と記す．また，＞は強単純化順序を表す．関数記号の順序付に関して，構成子は任意の演算子より小さいことが常に仮定されている．

【手続き 5.1】

0.P :=定義原理を満足する完備なTRS
  H0:=φ
  E0:=証明すべき等式の集合
  i:=0
1.if Ei=φ
    停止［定理成立］
  else
    EiからM=N を取り出す
  2.へすすむ

2.if M≡c(M1,...,Mn),N≡c(N1,...,Nn), c ∈ C
    Ri+1:=Ri,
    Ei+1:=Ei+{Mj=Nj 1≦j ≦n},
    i:=i+1とし1.へ戻る
  elif M≡c1(M1,...,Mm), N≡c2(N1,...,Nn), c1,c2∈ C, c1≠c2
    停止［定理不成立］
  elif M≡c(M1,...,Mn),c∈ C,N∈ V
    停止［定理不成立］
  elif M∈ V,N≡c(N1,...,Nn),c∈ C
    停止［定理不成立］
  elif M,N∈ V,M≠ N
    停止［定理不成立］
  elif M＞N
    F:=(M→N) とし3.へすすむ
  elif M＜N
    F:=(N→M) とし3.へすすむ
  else
    F:=(M↔N) とし3.へすすむ

3.if Pの規則とF との間に要対が存在しない
    停止［証明失敗］

if P中の規則とt との間に再帰対が存在
　ユーザの判断で補題を獲得
　Ei+1:=Ei+{補題}+{FとP 間の要対}
else
　Ei+1:=Ei+{FとP 間の要対}
if　Riに向付自由書換え規則 t→u が存在
　Ei+1:=Ei+1+{F と t→u 間の狭化式}
Ri+1:=Ri+{F},
i:=i+1とし4.へすすむ

4.repeat
　Eiの任意の等式t=u について
　if t=uがP+Riによって可約
　　if t↓ ≡ u↓
　　Ei:=Ei-{t=u}
　　else
　　Ei:=Ei-{t=u}+{t↓=u↓}
　if t=uがEiの他の等式または書換え規則
　　によって包含される
　　Ei:=Ei-{t=u}
until 上記のようなt=u が存在しない
1.へ戻る□

1.における等式の選択を公平(fair)にするために，次の選択戦略を用いる．
Ei={l1=r1,...,ln=rn}に対して，
min{max{:l1:,:r1:},...,max{:ln:,:rn:}}
にて選ばれる等式を選択する．ここで，:t: は項t に現れる関数記号数を表し，max,min は，各々要素の最大，最小値を返す関数である．これによって等式が一意に決まらない場合，次が用いられる．
min{min{:l1:,:r1:},...,min{:ln:,:rn:}}
これによっても等式が一意に決まらない場合，候補の中から (1)向付可能な等式，(2)向付不能な等式の順で任意に選択する．

通常の完備化手続きで行われている，書換え規則の簡約は，ここでは行わない．これは，潜在帰納法においては，規則の簡約が発生しにくく，検査の負荷の方が大きいとの判断によるものである．

3.におけるP の規則と「との間の要対は，FにP の規則を重ねるもののみ考える．しかも，複数の重複が可能な場合，F上唯一箇所の重複出現を選択する．

Ri中に向付自由書換え規則 t→u が存在する場合，3.において，Fと t→u 間の狭化が行われるが，これも，Fに対する t→u の適用のみ考える．

3.における要対は Fと Pとの間のみで考えるが，4.における簡約には，P+Riの書換え規則を利用する．これは，Fribourgの制限によるものである．等式包含の検査には，Eiの等式の他に，（もし存在すれば）Ri中の向付自由書換え規則も使われる．

6．実行例

6.1.加算の交換律

自然数をs とO により表現した場合，加算は書換え規則でP のように表される．この計算が自然数上で交換律(X+Y=Y+X) を満すことを証明する．演算子Dを{+}，構成子Cを{s,0} として +を正順辞書式部分項順序の関数，関数記号間の順序を0<s<+ とすると，Pは完備であり，かつ，定義原理を満足する．

```
P ={X+0 →X,                                  (r1)
    X+s(Y)→s(X+Y)}                           (r2)
R0= φ
E0={X+Y=Y+X}                                 (e1)
```

手続きを開始しE0から(e1)を選択すると，これは向付不能等式であるので，Fとして向付自由書換え規則，

```
  X+Y →Y+X                                   (r3)
```

要対として，

```
{s(X+Y)=s(Y)+X,                              (e2)
 X=0+X}                                      (e3)
```

が得られる．ここで，(e2)は(r3)に(r2)を，(e3)は(r3)に(r1)を重ねた要対である．

得られた等式や，E0から(e1)を取り除いた残りの等式の中に，P+R0+{F}によって簡約されるものはない．
これによりR1，E1は，

```
R1={(r3)}
E1={s(X+Y)=s(Y)+X,                           (e2)
    X=0+X}                                   (e3)
```

となる．次に，E1から(e3)を選択すると，右辺

＞方辺により Fとして，

0+X →X　　(r4)

要対として，

{s(0+X)=s(X),　　(e4)

0=0}　　(e5)

狭化により，

{X+0=X}　　(e6)

が得られる．ここで，(e4)は(r4)に(r2)を，(e5)は(r4)に(r1)を重ねた要対で，(e6)は(r4)を(r3)で狭化したものである．

得られた等式と，E1から(e3)を取り除いた残りの等式を，P+R1+{F}によって簡約すると，(e4)は，s(X)＝s(X)へ簡約され，削除される．(e5)は，すでに0 ＝0 となっており削除される．(e6)は，0 ＝0 へ簡約され，削除される．

これにより，R2，E2は，

R2=R1+{(r4)}

E2={s(X+Y)=s(Y)+X}　　(e2)

となる．E2から(e2)を選択すると，これは向付不能式であるので，Fとして，

s(X+Y)↔s(Y)+X　　(r5)

要対として，

{s(s(X+Y))=s(s(Y)+X),　　(e7)

s(X)=s(0)+X,　　(e8)

s(s(X)+Y)=s(s(Y)+X),　　(e9)

s(X)=s(0+X)}　　(e10)

狭化により，

{s(Y+X)=s(Y)+X,　　(e11)

s(X+Y)=X+s(Y)}　　(e12)

が得られる．ここで，(e7)は(r5)に(r2)を，(e8)は(r5)に(r1)を，(e9)は逆向(r5)に対し(r2)を，(e10)は同(r5)に(r1)を重ねた要対であり，(e11)(e12)は(r5)を(r3)で狭化したものである．

得られた等式と，E2から(e2)を取り除いた残りの等式を，P+R2+{F}によって簡約すると，(e10) は，s(X)＝s(X)へ簡約され削除される．(e12) は，s(X+Y)＝s(X+Y)となり，削除される．

これにより，R3，E3は，

R3=R2+{(r5)}

E3={s(s(X+Y))=s(s(Y)+X),　　(e7)

s(X)=s(0)+X,　　(e8)

s(s(X)+Y)=s(s(Y)+X),　　(e9)

s(Y+X)=s(Y)+X}　　(e11)

となる．E3から(e11) を選択すると，右辺＞左辺により Fとして，

s(Y)+X→s(Y+X)　　(r6)

要対として，

{s(s(X)+Y)=s(X+s(Y)),　　(e13)

s(X)=s(X+0)}　　(e14)

狭化により，

{X+s(Y)=s(Y+X),　　(e15)

s(Y)+X=s(X+Y),　　(e16)

s(X+Y)=s(Y+X),　　(e17)

s(Y)+X=s(X)+Y}　　(e18)

が得られる．ここで，(e13)は(r6)に(r2)を，(e14) は(r6)に(r1)を重ねた要対で，(e15)(e16) は(r6)を(r3)で，(e17)(e18)は(r6)を(r5)で狭化したものである．

得られた等式と，E3から(e11) を取り除いた残りの等式を，P+R3+{F}によって簡約すると，(e7)(e9)は，s(s(X+Y))=s(s(Y+X)) となり，(r3)により包含され，削除される．

(e8)は，s(X)＝s(X)となり，削除される．

(e13) は，s(s(X+Y)) ＝s(s(X+Y)) となり削除．

(e14) は，s(X)＝s(X)となり，削除．

(e15)(e16)(e17)(e18)は，s(X+Y)=s(Y+X) となり，(r3)により包含され削除される．

これにより，R4，E4は，

R4=R3+{(r6)}

E4= φ

ここで，手続きは停止し，交換律の成立が確認される．システムの最終状態は，以下の通りである．

P ={X+0 →X,　　(r1)

X+s(Y)→s(X+Y)}　　(r2)

R4={X+Y ↔Y+X,　　(r3)

0+X →X,　　(r4)

s(X+Y)↔s(Y)+X　　(r5)

s(Y)+X→s(Y+X)}　　(r6)

E4= φ

この例では，従来の方法で扱いの難しかった交換律を，向付自由書換え規則の導入により比較的効率良く証明できた．次に，項書換え型プログラムの検証例を示す．

6.2.2つのreverse

異る定義を与えられたリストの反転プログラムが，始モデル上で等価であることを証明する．以下のように，Pとして書換え規則による2種類のreverse の定義を与え，R0を空集合，E0を2つの定義が等価であることを示す等式の集合とする．演算子Dを{rev,app,nrev}，構成子Cを{[],[:]}とし，関数記号はすべて正順辞書式部分項順序の関数，関数記号間の順序は[]<[:]<nrev<app<revとすると，このP は完備であり，かつ，定義原理を満足する．

```
P ={rev([]) →[],                                  (r1)
    rev([X:Y])→app(rev(Y),[X])                    (r2)
    app([],X) →X,                                 (r3)
    app([X:Y],Z)  →[X:app(Y,Z)],                  (r4)
    nrev([],X)→X,                                 (r5)
    nrev([X:Y],Z) →nrev(Y,[X:Z]))                 (r6)
R0= φ
E0={rev(X) = nrev(X,[])}                          (e1)
```

手続きを開始すると，E0から等式(e1)が選択される．左辺＞右辺により Fとして，

```
  rev(X)→nrev(X,[])                               (r7)
```

要対として，

```
{nrev([],[])=[],                                  (e2)
 nrev([X:Y],[])=app(rev(Y),[X])}                  (e3)
```

が得られる．ここで，(e2)は(r7)に(r1)を，(e3)は(r7)に(r2)を重ねた要対である．

得られた要対と，E0から(e1)を取り除いた残りの等式を，P+R0+{F}によって簡約すると，(e2)は，[]=[]となり削除される．
(e3)は，nrev(Y,[X])=app(nrev(Y,[]),[X]) へ簡約される．
これによりR1，E1は，

```
R1={(r7)}
E1={nrev(Y,[X])=app(nrev(Y,[]),[X])}              (e3)
```

となる．次に，E1から(e3)を選択すると，右辺＞左辺により Fとして，

```
  app(nrev(Y,[]),[X])=nrev(Y,[X])                 (r8)
```

要対として，

```
{app([],[X])=nrev([],[X]),                        (e4)
 app(nrev(X,[Y]),[Z])=nrev([Y:X],[Z])}            (e5)
```

が得られる．ここで，(e4)は(r8)に(r5)を，(e5)は(r8)に(r6)を重ねた要対である．

得られた要対と，E1から(e3)を取り除いた残りの等式を，P+R1+{F}によって簡約すると，
(e4)は，[]=[]となり削除される．
(e5)は，app(nrev(X,[Y]),[Z])=nrev(X,[Y,Z]) へ簡約される．

ここで，(e5)は(r8)と(r6)の再帰対になっており，無限実行の可能性が検出される．

再帰対の構造より，要対の一般形は，

```
  app(nrev(X,[Y1,...,Yn]),[Z])
                  =nrev([Yn,...,Y1:X],[Z])
```

であることがわかる．これは，Pによって，次の一般形へ簡約される．

```
  app(nrev(X,[Y1,...,Yn]),[Z])
                    =nrev(X,[Y1,...,Yn,Z])
```

これをさらに一般化する補題としては，次のものが考えられる．

```
  app(nrev(X,Y),Z)=nrev(X,app(Y,Z))               (e6)
```

そこで，これを補題とし，併せてE2へ追加する．これによりE2，R2は，

```
R2=R1+{(r8)}
E2={app(nrev(X,[Y]),[Z])=nrev([Y:X],[Z]),
                                                  (e5)
    app(nrev(X,Y),Z) = nrev(X,app(Y,Z))}
                                                  (e6)
```

となる．E2から(e6)を選択すると，左辺＞右辺により Fとして，

```
  app(nrev(X,Y),Z)→nrev(X,app(Y,Z))               (r9)
```

要対として，

```
{app(Y,Z)=nrev([],app(Y,Z)),                      (e7)
 app(nrev(X,[Y:Z],V)=nrev([Y:X],app(Z,V))}
                                                  (e8)
```

が得られる．ここに，(e7)は(r9)に(r5)を，(e8)は(r9)に(r6)を重ねたものである．

得られた要対と，E2から(e6)を取り除いた残

りの等式を，P+R2+{f}によって簡約すると，
(e5)は，nrev(X,[Y,Z]) ＝nrev(X,[Y,Z]) となり削除される．
(e7)も，app(Y,Z)＝app(Y,Z)となり削除される．
(e8)も，nrev(X,[Y:app(Z,V)])＝
nrev(X,[Y:app(Z,V)]) となり削除される．
これにより，R3，E3は，

R3=R2+{(r9)}
E3= φ

となる．これにより手続きは停止し，最初に与えた等式が，帰納的定理であることが確認される．システムの最終状態は，以下の通りである．

```
P ={rev([]) →[],                          (r1)
    rev([X:Y])→app(rev(Y),[X])            (r2)
    app([],X) →X,                         (r3)
    app([X:Y],Z)  →[X,app(Y,Z)],          (r4)
    nrev([],X)→X,                         (r5)
    nrev([X:Y],Z) →nrev(Y,[X:Z])}         (r6)
R3={rev(X)→nrev(X,[]),                    (r7)
    app(nrev(Y,[]),[X]) →nrev(Y,[X]),(r8)
    app(nrev(X,Y),Z)→nrev(X,app(Y,Z))}
                                          (r9)
E3= φ
```

7．おわりに

Metis上の潜在帰納法について説明した．従来の手続きには，①結果（成立・不成立）と共に停止する②等式の向付に失敗して停止する③停止しないの３つの場合があったのに対し，我々の方法は，(1)向付自由規則の導入によって②の場合を防ぎ，また，(2)無限実行の検出基準により，ユーザの介入によって③の多くの場合を回避することにも成功した．

今後はさらに定理証明能力の向上を目指していく予定であるが，今のところ以下の課題について検討している．

(1) 基礎合流性判定方法の検討
(2) 無限実行検出条件の強化
(3) 補題の自動獲得メカニズムの研究

謝辞

本研究は第五世代コンピュータプロジェクト・知的プログラミングシステム研究，開発の一環として行われたものである．研究の機会をくださった，（財）新世代コンピュータ技術開発機構の渕所長，同古川次長，同長谷川第１研究室室長，同伊藤第３研究室室長，また多くの議論をしていただいた第１研究室のメンバー，旧TRS-WGのメンバーの方々に心より感謝致します．

参考文献

[Dershowitz 84] Dershowitz,N.: "Termination",Rewriting Techniques and Applications,INCS 202(1985),PP.180-224

[Fribourg 86] Fribourg,L.:"A strong restriction of the inductive completion procedure",13th International Colloquium on Automata,Languages and Programming, LNCS 226(1986),pp.105-115.

[Goguen 80] Goguen,J.A.:"How to prove algebraic induction hypothesis without induction, witha application to the correctness of data type implementation" ,5th Conference on Automated Deduction, LNCS 87(1980),pp.356-373.

[Hermann 85] Hermann,M.and Privara,I.: "On nontermination of Knuth-Bendix algorithm",Research Report VUSEI-AR-OPS-3/85,Institute of Socio-Economic Information and Automation,CS-842 21, (1985).

[Hsiang 87] Hsiang,J.:"On word problems in equational theories",Automata Language and Programmig,LNCS-267(1987), pp.54-71.

[Huet 80] Huet,G.and Oppen,D.C.: "Equations and rewrite rules : A survey" ,Formal Language: Perspectives and Open Problems,(R.Book,eds.), Academic Press (1980),pp.349-405.

[Huet 82] Huet,G.and Hullot,J.M.:"Proofs by induction in equational theories with constructors",J.Comput.Syst.Sci.,Vol.25, No.2(1982),pp.239-266.

[Jouannaud 84] Jouannaud,J.P.and Kirchner

,H.: "Completion of a set of rules modulo a set of equations",11th ACM Symposium on Principle of Programming Languages(1984),pp.83-92.

[Jouannaud 85] Jouannaud,J.P.and Kounails,E.: "Proofs by Induction in Equational Theories without Constructors ",Bulletin of EATCS 27(1985),pp.49-55.

[Knuth 70] Knuth,D.E.and Bendix,P.B.: "Simple Word Problems in Universal Algebras",Computational Problems in Abstract Algebra (J.Leech eds.),Pergamon Press,Oxford(1970),pp.263-297.

[Husser 80] Husser,D.R.:"On proving inductive propertiesof abstract data types",7th ACM Symposium on Principle of Programming Language(1980),pp.154-162.

[Ohsuga 86] Ohsuga,A.and Sakai,K.:"Metis: A Term Rewriting System generator",RIHS Symposium on Software Science and Engineering(1986).

[大須賀 87] 大須賀昭彦："Metisユーザーズマニュアル",ICOT TR,(発行予定).

[Peterson 81] Peterson,G.E.andStickel,M.: "Complete sets of reductions for equational theories with complete unification algorithms",J.ACM,Vol28 (1981),pp.233-264

[Sakai 84a] Sakai,K.:"An Ordering method for term rewriting systems",ICOT TR-062 (1984).

[Sakai 84b] Sakai,K.:"Knuth-Bendix Algorithm for Thue System Based on Kachinuki Ordering",ICOT TM-0087(1984).

[坂井 87] 坂井公:"Knuth-Bendixの完備化手続きとその応用",コンピュータソフトウェア,Vol4,No.1(1987),pp.2-22.