TM-0972

モデル推論

石坂　裕毅、有川　節夫（九州大学）

November, 1990

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

モデル推論

石坂裕毅，有川節夫

1 はじめに

次のような論理プログラムを考えてみよう．

$$rev([x|y],z) \leftarrow rev(y,y_1), con(x,y_1,z).$$
$$rev([],[]).$$
$$con(x,[y|y_1],[y|z]) \leftarrow con(x,y_1,z).$$
$$con(x,[],[x]).$$

このプログラムから導出(証明)できるグランドアトムの集合Mは以下のようになる．

$\{rev([],[]), rev([a],[a]), rev([a,b],[b,a]), \ldots,$
$con(a,[],[a]), con(a,[b],[b,a]), con(a,[a,b],[a,b,a]), \ldots\}$

上のプログラムが分かっていれば，任意のグランドアトムαに対して，αのMにおける所属性(membership)を判定することができる．モデル推論とは，この逆，すなわち，任意のαに対して，αのMにおける所属性が判定できるという仮定のもとで，上のようなプログラムを推論するという一種の帰納推論である．Shapiroはこの問題を一階述語論理を対象として形式化するとともに，それを解くための極めて有効なアルゴリズムを示した[16]．

本解説では，Shapiroによるモデル推論について紹介し，モデル推論のもつ問題点，特に，理論名辞の生成問題に関連する最近の研究のいくつかについて概観する．

2 モデル推論アルゴリズム

本節では，モデル推論問題をより形式的に定義し，それを解く具体的なアルゴリズムとして，もっとも単純な枚挙型アルゴリズムとShapiroが与えた漸増的アルゴリズムの基本的なアイディアについて述べ，両者の効率の違いを見てみる．なお，以下では，簡単のために，推論対象を論理プログラムの世界に制限して説明を行なう．論理プログラムに関する未定義用語については参考文献[12]を参照されたい．

2.1 モデル推論問題

有限個の述語記号と関数記号(定数記号は0引数関数記号とみなす)をもつ1階言語をLとする．L上の論理プログラム(以下では，単にプログラムと呼ぶ)とは，Lから構成される次のような確定節(definite clause)の有限集合である．ただし，A, B_iはアトム(atomic formula)である．

$$A \leftarrow B_1, B_2, \ldots, B_n. \quad (n \geq 0)$$

以下では，確定節のことをプログラム節あるいは単に節と呼び，Aを節の頭部，$B_1, \ldots, B_n$の部分を節の本体と呼ぶ．$n = 0$であるような節を単一節(unit clause)と呼ぶ．Lから構成可能なすべてのプログラム節の集合を$\mathcal{L}$で表し，$\mathcal{L}$の有限部分集合全体，すなわち，L上のプログラム全体の集合を$2^{\mathcal{L}}$で表す．$\mathcal{L}$および$2^{\mathcal{L}}$は帰納的に可算(recursively enumerable)である．

L上のHerbrand基底(base)を$\mathcal{B}_L$で表す．すなわち，$\mathcal{B}_L$はL中の述語記号と関数記号だけから構成されるグランドアトム全体の集合である．$\mathcal{B}_L$の任意の部分集合をL上のHerbrand解釈と呼ぶ(以下では，単に解釈と呼ぶ)．L上の解釈Mと$\mathcal{L}$の要素$C = A \leftarrow B_1, \ldots, B_n$ $(n \geq 0)$および$\mathcal{B}_L$の要素αに対し，

$$\alpha = A\theta \text{ かつ } B_i\theta \in M \quad (1 \leq i \leq n)$$

を満たすような代入(substitution)θが存在するとき，節Cは解釈Mに関してαを被覆するという．節Cが解釈Mに関して被覆するグランドアトム全体の集合を$C(M)$ $(\subseteq \mathcal{B}_L)$で表す．$C(M) \subseteq M$のとき，節$C$は解釈$M$に関して真であるといい，それ以外のとき，偽であるという．プログラムP中のすべての節が解釈Mに関して真であるとき，MはPのモデルであるという．プログラムPのすべてのモデルの集合を$\{M_i\}_{i \in I}$とするとき，$\cap_{i \in I} M_i$，すなわち，Pのすべてのモデルの共通部分もまたPのモデルとなる．このモデルをPの最小Herbrandモデルと呼び，$M(P)$で表す．

プログラムPからアトムαが導出原理(resolution principle)によって導出可能であることを，$P \vdash \alpha$によって表す．また，$P \vdash_n \alpha$でnステップ以下で導出可能であることを，$P \not\vdash_n \alpha$でnステップでは導出不可能であることを表す．

グランドアトムαを与えると，αが解釈Mの要素であればTrueを，要素でないならばFalseを返すような仕掛けの存在を仮定する．この仕掛けのことを解釈Mに関するオラクルという．オラクルはMAT学習[1]における所属性質問(membership query)に答えられる教師に対応する．

モデル推論問題とは，一階言語LとL上の解釈Mに

[1]本特集「EXACT学習」等を参照されたい．

関するオラクルが与えられたとき，M をモデルとし，かつ，M の任意の要素 α に対して $P \vdash \alpha$ となるようなプログラム P を見つけることである．さて，ここまで $\mathcal{B}_L$ の部分集合のことを解釈と呼んできたが，モデル推論問題においては，対象とする解釈は結果的にモデルとなることを意図されている．したがって，以下では解釈のことを直接モデルと呼ぶことにする．また，論理プログラムの性質として $M(P) = \{\alpha \in \mathcal{B}_L \mid P \vdash \alpha\}$ が成り立つから，この問題は次のように言い換えることができる．

> 一階言語 L と L 上のモデル M に関するオラクルが与えられたとき，$M = M(P)$ なるプログラム P を見つける．

2.2 枚挙法によるアルゴリズム

まず，もっとも単純な枚挙法[5]に基づくアルゴリズムを見てみよう．グランドアトム $\alpha \in \mathcal{B}_L$ と，α をモデル M に関するオラクルに与えたときの出力 $V \in \{\mathrm{True}, \mathrm{False}\}$ との対 $\langle \alpha, V \rangle$ のことを M に関する事実(fact)と呼び，$V = \mathrm{True}$ であるような α を正事実，$V = \mathrm{False}$ であるような α を負事実と呼ぶ．$\mathcal{B}_L$ のすべての要素の帰納的枚挙 (recursive enumeration) $\alpha_1, \alpha_2, \ldots$ を M に関するオラクルに順次与えることによって得られる事実の列 $\langle \alpha_1, V_1 \rangle, \langle \alpha_2, V_2 \rangle, \ldots$ をモデル M の枚挙と呼ぶ．アルゴリズム1はこの M の枚挙を利用し，直接オラクルに質問をすることはない．

$2^{\mathcal{L}}$ のすべての要素の帰納的枚挙を一つ固定し，これを $P_1, P_2, \ldots$ とすると，以下のような単純なアルゴリズムが考えられる．すなわち，可能なプログラムすべてを枚挙し，それまで受け取った正負の事実と突き合わせることにより，$M = M(P)$ なる P を探索するのである．

アルゴリズム1

$k := 1$; $S_{\mathrm{True}} := \{\ \}$; $S_{\mathrm{False}} := \{\ \}$;
repeat
  read the next fact $F_n = \langle \alpha, V \rangle$;
  $S_V := S_V \cup \{\alpha\}$;
  while $\exists \beta \in S_{\mathrm{False}} :\ P_k \vdash_n \beta$ or
    $\exists \beta \in S_{\mathrm{True}} :\ P_k \not\vdash_{h(|\beta|)} \beta$ *do*
    $k := k + 1$
  output P_k
forever

一般に，各仮説と既知の事実との無矛盾性の判定，すなわち，$P_k \vdash (\forall)\beta$ の判定は一般に決定不能である．したがって，上のアルゴリズムでは，while ループの条件判定における計算のステップ数に対して，n と $h(|\beta|)$ という上限を設けている．ただし，n はそれまでに受け取った事実の個数であり，h はある計算可能関数，$|\beta|$ は β 中の関数記号の出現個数を表す．負事実に関しては，n ステップ以内で導出できれば矛盾であるとし，正事実に関しては，その正事実のサイズに依存したステップ数以内に導出できない場合，矛盾であると判断する．

さて，プログラム P は，ある計算可能関数 h と $P \vdash \alpha$ なるほとんどすべての(すなわち，有限個を除いたすべての) $\alpha \in \mathcal{B}_L$ に対して，

$$\min\{i \mid P \vdash_i \alpha\} \leq h(|\alpha|)$$

を満たすとき h 従順(h-easy)であるという．$M = M(P)$ なる h 従順なプログラム P が存在するとき，モデル M は h 従順であるという．そうすると，L 上の任意の h 従順なモデル M と M の任意の枚挙[2]に対して，アルゴリズム1は M を極限において同定する，すなわち，アルゴリズム1の出力列が $M = M(P)$ なるプログラム P に収束する[3]ことが保証される[16]．

2.3 漸増的枚挙アルゴリズム

明らかに，単純な枚挙法によるアルゴリズムは効率が悪い．その根本的な原因は，現在の推測(推論アルゴリズムの出力) P_k と次の推測の候補 P_{k+1} との間には，通常の枚挙ではなんらの関係もないということにある．つまり，推測の候補者(以後，仮説と呼ぶ)の枚挙は，これから推論しようとしているモデルに無関係に定まっており，また当然それまでにアルゴリズムが受け取った事実とも無関係である．そのため，たとえば現時点で正しい推測に非常に近いところまで到達していたとしても，次に検査する候補者は全く関係のないものであることもあるわけである．

Shapiro が与えた漸増的枚挙アルゴリズムでは，この問題点に対し，論理プログラムの単調性，すなわち，任意のプログラム P と任意の節 C に対して $M(P) \subseteq M(P \cup \{C\})$ なる性質を利用した改良を行なっている．つまり，現在の仮説 P に新たな節を加えたり，P から節を削除することによって $M(P)$ を増減させ，徐々に M に近づけるのである．もう少し具体的には，次の2つの場合に分かれる．

1. $\exists \beta \in M(P)$ s.t. $\beta \notin M$.
2. $\exists \beta \in M$ s.t. $\beta \notin M(P)$.

2の場合は，新たな節を P に加えることによって $M(P)$ を増加させる．このため，アルゴリズム2ではプログラム

[2] すなわち，M の事実を与える順番に依存しない．
[3] ある有限時点以降は常に P を出力する．

全体($2^{\mathcal{L}}$)の枚挙ではなく，節全体($\mathcal{L}$)の枚挙$C_1, C_2, \ldots$を用いている．1の場合には，Pから1つの節を削除することによって$M(P)$を減少させる．このとき，どの節を削除するかが問題となるが，それは，次の命題から明らかである．

> $M = M(P)$ならば，P中のすべての節はMに関して真である．

すなわち，Mに関して偽であるような節を含むPは$M \neq M(P)$であるから，Mに関して偽であるような節Cを削除すればよい．1の場合にそのような節が存在すること，およびそれを見つけるための手続きについては次節で述べる．また，節の枚挙$C_1, C_2, \ldots$が重複した節を含まないようにすることによって，一旦偽であることが判明した節が2度と仮説に含まれないようにできる．これが漸増的(incremental)と呼ばれる所以である．アルゴリズム2はアルゴリズム1と同等な能力をもつ．すなわち，任意のh従順なモデルを極限において同定することができる．[16, 17].

アルゴリズム2

$k := 1$; $P := \{\}$; $S_{\mathrm{True}} := \{\}$; $S_{\mathrm{False}} := \{\}$;
repeat
  read the next fact $F_n = \langle \alpha, V \rangle$;
  $S_V := S_V \cup \{\alpha\}$;
  repeat
    while $\exists \beta \in S_{\mathrm{False}} : P \vdash_n \beta$ *do*
      find and remove a false clause in P
    while $\exists \beta \in S_{\mathrm{True}} : P \not\vdash_{h(|\beta|)} \beta$ *do*
      $P := P \cup \{C_k\}$; $k := k + 1$
  until neither of the while loops is entered
  output P
forever

ここで，推論の途中で生成され，それまで受け取った事実との無矛盾性のテストが行なわれる仮説の個数をもとに，単純な枚挙法と漸増的枚挙法との効率の違いをみてみよう．

いま，仮に$\mathcal{L}$が有限集合で，その要素の個数をnと仮定してみる．単純な枚挙法では$\mathcal{L}$の部分集合全体を虱潰しに探索することになるから，最悪の場合2^n個の仮説を生成しテストすることになる．ところが，漸増的枚挙の場合，高々n回の節の付加と高々n回の節の削除が行なわれるだけである．すなわち，最悪の場合でも，高々$2n$個の仮説しか生成テストされないのである．逆にいえば，高々$2n$回の生成テストにより2^n個の仮説空間を虱潰しに探索したのと同じ効果が得られるのである．2^nと$2n$，似ているようだがその差は歴然としている．

3 矛盾点追跡と精密化

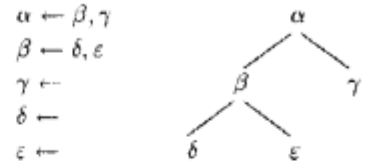
前節で述べたような効率化のためには，節の削除が適切に行なわれなければならない．Shapiroは，現在の仮説Pが負事実と矛盾した場合(アルゴリズム2の1番目のwhileループ)に，Pの中から偽である節を探し出すための手続きを与えている．さらに，節の枚挙の仕方自体にも工夫を凝らすことによって無駄な節の枚挙を省いている．ここでは漸増的枚挙アルゴリズムにとって核となるそれら2つの手法について述べる．

3.1 矛盾点追跡アルゴリズム

負事実を導出するような仮説から偽である節を探し出す手続きは矛盾点追跡(contradiction backtracing)アルゴリズムと呼ばれる．負事実を導出するプログラム中に偽である節が存在することは，以下のアルゴリズムの説明から明らかであろう．

ある負事実$\alpha \notin M$に対して，$P \vdash_n \alpha$であると仮定する．このとき，その導出に対応するトレース木が得られる．矛盾点追跡アルゴリズムは，このトレース木を入力とし，オラクルに質問を行ないながら，P中の節でMにおいて偽なものを探し出す．たとえば，図1の右側にあるようなトレース木が与えられたと仮定しよう．図1の左側に挙げた節は，このトレースに関連しているP中の節のグランドインスタンスである．

図 1: 矛盾点追跡アルゴリズムへの入力



まず，アルゴリズムはαの子β, γに対して，それらがMの要素かどうかをオラクルに尋ねる．β, γがともにMの要素であるならば，$\alpha \leftarrow \beta, \gamma$をインスタンスとする$P$中の節を返す．すなわち，ある代入$\theta$に対して，$A\theta = \alpha \notin M$かつ$B\theta = \beta \in M, C\theta = \gamma \in M$であるから節$A \leftarrow B, C \in P$は$M$において偽ということになる．$\gamma$が$M$の要素でないならば，即座に，$\gamma \leftarrow$をインスタンスとする節が偽であることが分かる．$\beta$が$M$の要素でない場合には，$\beta$を根とする部分木を入力として再帰的に探索を続ける．

導出$P \vdash_n \alpha$に対応するトレース木は有限であり，最終的には葉の部分に対応する単一節にまで到達するわけだから，この手続きによって必ずMにおいて偽である節を探し出すことができる．基本的に，与えられたトレース木の各ノードに対して，それがMの要素かど

うかをオラクルに質問するだけであるから，大きな計算コストを必要としない．

2^n と $2n$ の差を実現している本質的な要素はこの矛盾点追跡アルゴリズムの存在にある．実際，このアルゴリズムを利用することによって，いくつかの形式言語のクラスが多項式時間学習可能であることが示されている [1, 7, 8]．また，アルゴリズム1ではオラクルに直接質問することはなかったが，矛盾点追跡アルゴリズムでは必要な情報(ここでは，グランドアトム α の M における所属性)を得るために質問がなされる．このことは，推論(学習)の効率化における所属性質問の重要性を示唆している．推論過程において利用できる質問の種類と，推論可能性およびその効率との関係については本特集の他の解説や参考文献 [2] を参照されたい．

3.2 精密化子による節の枚挙

アルゴリズム2で用いられる節の枚挙は，網羅的($\mathcal{L}$ の要素がすべて出現する)であれば一応アルゴリズムの完全性は保証される．重複がなければ，同じ過ちを二度繰り返さない分だけ効率的になる．しかし，実際には，さらに無駄な節の枚挙を抑えつつ網羅的な枚挙を保証することが可能である．

たとえば，推論のある時点で節 C がモデル M に関して偽であることが判明したとする．すると，$C(M) \subseteq C'(M)$ であるようなすべての節 C' は M に関して偽であるから，それ以降の節の枚挙に登場する必要はない．逆に，任意のモデル M に関して $C'(M) \subseteq C(M)$ であるような節 C が仮説 P 中に存在する場合，C が偽であることが判明しない限りは，C' を P に付加する必要はない[4]．このような性質を節の枚挙に反映させるのが精密化子(refinement operator)の役割である．以下では，$C'(M) \subseteq C(M)$ という関係を，より多くのグランドアトムを被覆するという意味で C の方が C' よりも一般的である(C' は C よりも具体的である)と呼ぶことにする．

精密化子 ρ は，任意のモデル M に対して次の条件を満たすような節から節の有限集合への写像である．

$$\forall C' \in \rho(C),\ C'(M) \subseteq C(M)$$

精密化子による節の枚挙とは，もっとも一般的な節からスタートし，ρ を繰り返し適用することによって，徐々に具体的な節を生成するようなものである．もう少し具体的には，矛盾点追跡アルゴリズムによって偽であると判定された節だけに適用を繰り返す．これによって先ほど述べたような無駄な節の枚挙を抑えることができるのである．

ここで，ρ の具体例をみてみよう．任意のモデル M に対して $C(M) = \mathcal{B}_L$ なるもっとも一般的な節を $\square$ で表すことにする．次のうちどれかが成立するとき $C' \in \rho(C)$ であると定義する．

1. $C = \square$ のとき $C' = a(x_1, \ldots, x_n)$．ただし，a は n 変数の述語記号で x_i は互いに異なる変数記号．
2. $C' = C\theta$．ただし，θ は C 中の相異なる2つの変数を単一化するような代入．
3. $C' = C\theta$．ただし，θ は C 中のある変数を，n 変数関数記号 f と C に現れない互いに異なる変数 x_i からなる項 $f(x_1, \ldots, x_n)$ で置き換えるような代入．
4. $C = A \leftarrow B_1, \ldots, B_m$ のとき $C' = A \leftarrow B_1, \ldots, B_m, a(x_1, \ldots, x_n)$．ただし，$a$ は n 変数の述語記号で x_i は互いに異なる変数記号．

$\square$ からスタートし，この ρ を繰り返し適用することによって $\mathcal{L}$ のすべての要素を生成できる．Shaprio はこの他にも，(確定節に限らない)一般の一階論理における節全体を生成できる精密化子や，ある特殊なクラスだけを生成するような精密化子の例をいくつか与えている [17, 16]．

4 最近の研究動向

矛盾点追跡と精密化による漸増的枚挙は，モデル推論問題を解くための極めて巧妙かつ有効な手法であるといえる．しかしながら，モデル推論問題自体は一般的な例からの概念学習やプログラム合成の問題とは異なる側面をもっており，そのために，モデル推論アルゴリズムを直接一般の概念学習に適用することは困難である．つまり，モデル推論問題においては，予め与えられた一階言語 L で表現可能なモデルだけが推論対象となっているが，そのような言語(しかも，効率を考えるならば，できるだけコンパクトなもの)を予め用意することや，推論対象として全体的なモデルを想定することは，一般の概念学習等においては不自然な設定となるのである．

たとえば，1,0から成る回文の概念を例から学習するような場合，自然な設定としては，$pal(\_)$ という適当な述語を用意して，

$$M = \{pal([0]), pal([1]), pal([0,1,0]), pal([1,0,1]), \ldots\}$$

をモデルとするようなプログラムを生成してくれることが望まれる．しかし，そのようなプログラムを $pal(\_)$ だけで記述することは不可能であり，以下のような $reverse(\_,\_)$ 等の補助的な述語が必要であることが知られている [16]．

$$pal(x) \leftarrow reverse(x, x).$$

したがって，モデル推論によって $pal(\_)$ に対する正しいプログラムを推論するためには，$pal(\_)$ 以外の補助的な

[4] C' を用いて導出されるグランドアトムはすべて C を使って導出可能である．

述語，たとえば $reverse(\_,\_)$ を含む一階言語が予め与えられなければならない．さらに $reverse(\_,\_)$ が自分以外の補助的述語を必要とする場合には，その述語も含まれなければならない．また，矛盾点追跡アルゴリズムを実行するために，教師としてのオラクルは $pal(\_)$ に関する情報だけでなく $reverse(\_,\_)$ 等の補助的述語に関する情報も与えることができなければならないのである．

$reverse(\_,\_)$ のように，目標となる概念（ここでは $pal(\_)$）を定義するための補助的な述語のことを理論名辞(theoretical term)と呼ぶ．モデル推論における理論名辞を含む一階言語の仮定は，一般の概念学習の立場から常に指摘されてきた問題点である．理論名辞の生成・発見の問題は，機械学習にとって極めて本質的であり，避けて通るべき問題ではないが，もっとも困難な問題の一つであることも確かである．以下では，この問題に関連する最近の研究[14, 13, 4, 11, 10, 7, 21, 8, 19, 20]の中からいくつかを紹介しておく．

4.1 逆導出

Muggleton と Buntine[14, 13] は逆導出(inverting resolution)と呼ばれる導出原理のちょうど逆の操作を行なう手法を提案し，それに基づいたシステムCIGOLを試作している．逆導出は基本的にtruncation, absorption, intra-construction[5]の3つの操作からなる．CIGOLはそれらの操作を繰り返し適用することによって，与えられた正事実から構成的に仮説を生成する．truncationはPlotkinの最小汎化(least generalization)[15]を利用して正事実の集合から単一節を生成し，absorptionは既存の述語だけから非単一節を生成する．新たな述語の生成はintra-constructionによって行なわれる．

truncationすなわちPlotkinの最小汎化は，たとえば，

$$arch([block], beam, [block])$$
$$arch([blick], beam, [blick])$$

という2つのアトムに対し，$arch([x], beam, [x])$ のように，それらの共通部分をできるだけ多く残した形で，それらを一般化したアトムを得る操作である．

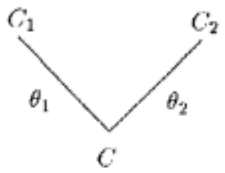
absorptionは目標となるプログラム上でのある導出過程において，図2のような1ステップの導出が存在するとものと想定し，C と C_1 から逆に C_2 を推測するような操作である．ただし，

$$C = (C_1 - \{L_1\})\theta_1 \cup (C_2 - \{L_2\})\theta_2$$

とし，L_1 は C_1 中の正リテラル L_2 は C_2 中の負リテラル[6]で $\overline{L_1}\theta_1 = L_2\theta_2$ とする（$\overline{L_1}$ は L_1 の否定を表す）．

ここで C_1 が単一節 $\{L_1\}$ であると仮定して，上の式

図 2: 1ステップの導出



を C_2 について代数的に解くと

$$C_2 = C\theta_2^{-1} \cup \{L_2\}$$

を得る．ただし，θ_2^{-1} は θ_2 の逆代入(inverse substitution)と呼ばれるもので，θ_2 によって項に置き換えられた部分を元の変数に戻す操作である[7]．また，$L_2 = \overline{L_1}\theta_1\theta_2^{-1}$ と表せるから，結果的に

$$C_2 = (C \cup \{\overline{L_1}\}\theta_1)\theta_2^{-1}$$

となる．Muggletonは，与えられた C と C_1 から θ_1 と θ_2^{-1} を非決定的に求めることによってabsorptionを行うアルゴリズムを与えている．

たとえば，$C = \{u < s(s(u))\}$，$C_1 = \{v < s(v)\}$ に対してabsorptionを適用することによって，

$$x < y \leftarrow s(x) < y.$$

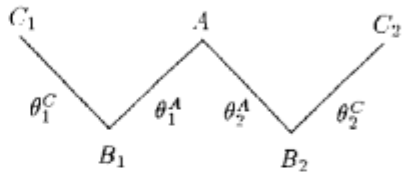
のような節 C_2 を生成することができる．

intra-constructionは目標のプログラム上で図3にあるような導出が可能であると想定して，B_1 と B_2 から A を推測する[8]操作である．ただし，

$$B_i = (A - \{L\})\theta_i^A \cup (C_i - \{L_i\})\theta_i^C$$

とし，L は正リテラル L_i は負リテラルとする[9]．

図 3: 共通の節に対する2つの導出



abosroptionの場合と同様に $C_i = \{L_i\}$ であると仮定すると次のような関係を得る．

$$B_i = (A - \{L\})\theta_i^A$$

したがって，$(A - \{L\})$ は各 B_i をインスタンスとするような節ということになる．そのような節は有限個しかないから，その中の一つを固定し B とすると，各 θ_i^A を決定することができ，また，

$$A = B \cup \{L\}$$

[5] absorption, intra-construction に関しては，各々に対する相対の操作 identification と inter-construction が存在する．

[6] 正負が逆の場合の操作は identification と呼ばれる．

[7] θ^{-1} は θ に依存して決まるが一意に定まるものではない．

[8] 実際には，C_1, C_2 も推測される．

[9] 正負が逆の場合の操作は inter-construction と呼ばれる．

と表すことができる．このL(Lは負リテラルであるから，正確には$\overline{L}$)が新たに生成される理論名辞ということになる．intra-constructionでは各θ_i^Aによって置き換えられるB中の変数の中から，あるヒューリスティクスに基づいて選択された変数$\{v_1,\ldots,v_m\}$と適当な文字列pによって$p(v_1,\ldots,v_m)$なる理論名辞の導入が行われる．

たとえば，

$$B_1 = min(x_1,[s(x_1)|y_1]) \leftarrow min(x_1,y_1).$$
$$B_2 = min(x_2,[s(s(x_2))|y_2]) \leftarrow min(x_2,y_2).$$

に対して

$$B = min(x,[y|z]) \leftarrow min(x,z).$$

とすると，

$$\theta_1^A = \{x/x_1, y/s(x_1), z/y_1\}$$
$$\theta_2^A = \{x/x_2, y/s(s(x_2)), z/y_2\}$$

となり，その中の変数xとyを選ぶことによって新たな述語$p(x,y)$を導入し，

$$min(x,[y|z]) \leftarrow min(x,z), p(x,y).$$

のような節を生成できる．

CIGOLは，教師から与えられた正事実に対して，truncation, aborption, intra-constructionを適用することによって構成的に仮説を生成する．したがって，Shapiroのモデル推論アルゴリズムにあるような枚挙的要素がなく，より効率的なシステムとなっている．また，intra-constructionによる補助述語の導入によって，より柔軟な学習過程を実現しているといえる．実際，CIGOLによって，リストの反転プログラムやリスト中の最小要素を求めるプログラムの合成の途中で，それらのプログラムにおいて補助的な述語であるリストの連接関係(append)や大小関係(<)が生成された例も報告されている．しかし，absorptionやintra-constructionにおける本質的な部分においてヒューリスティクスに頼らざるを得ない点や，教師に対してかなり高い能力を仮定している[10]点等，問題点があることも事実である．特に，intra-constructionによって生成された新しい述語，たとえば，先の例にあるような述語$p(_,_)$を定義する節を生成する際に，CIGOLは，pに対する適当な名前や$p(_,_)$に関する新たな事実を教師に要求する．すなわち，生成された述語の意味は教師によって決定されることになり，理論名辞の生成問題に対して完全な解決を与えているとは言い難い．

4.2 Lingの構成的アルゴリズム

Ling[11, 10]の構成的推論アルゴリズムは，Summers[18]によるトレースからのLispプログラム合成の手法に基づいており，Banerji[4]が提案した理論名辞生成のアイデアを自然な形で実現できる枠組を与えている．

Banerjiのアイディアとは，ある時点での仮説の中に，同じ頭部をもち本体の一部を共有しているような2つの節$P \leftarrow A,D$, $P \leftarrow B,D$が存在するとき，それらを以下のような3つの節で置き換えることによって，新たな述語$NewP(_,\ldots,_)$を導入するものである．

$$P \leftarrow NewP(t_1,\ldots,t_n), D.$$
$$NewP(x_1,\ldots,x_n) \leftarrow A'.$$
$$NewP(x_1,\ldots,x_n) \leftarrow B'$$

ただし，$t_1,\ldots,t_n$はA,Bに現れるすべての項であり，A',B'はA,B中の項t_iを変数x_iで置き換えたものである．この変換で生成される仮説自体はもとの仮説と等価なものである．しかし，新たに導入された節に対して，以下で述べるような一般化則を適用することによって仮説の一般化が行える場合もある．また，新しく導入された述語$NewP$はA',B'によって定義されているから，意味をもった述語として生成されており，CIGOLに見られるような問題は起こらない．

Lingのアルゴリズムは，SummersのTHESYS[18]と同様に，与えられた正事実から論理トレース(logical trace)と呼ばれる節を生成する．たとえば，リストの反転を行う述語$rev(_,_)$に関する正事実$rev([a,b],[b,a])$に対し，

$$rev(x,y) \leftarrow car(x,x_1), a(x_1), cdr(x,x2),$$
$$car(x_2,x_3), b(x_3), cdr(x_2,x_4), nil(x_4),$$
$$car(y,y_1), b(y_1), cdr(y,y_2), car(y_2,y_3),$$
$$a(y_3), cdr(y_2,y_4), nil(y_4)$$

のような節を生成し，仮説に付加する．$car(x,y), cdr(x,y)$はそれぞれLispにおける関数CAR, CDRに対応しており，$\mathrm{CAR}(x)=y, \mathrm{CDR}(x)=y$という関係を表す．$a(x), b(x), nil(x)$はそれぞれ$x=a, x=b, x=nil$を表す述語である．すなわち，論理トレースとは，与えられた正事実に現れているデータの構造を本体部で忠実に記述しているような非単一節であり，その正事実のみを被覆するという性質をもつ．car, cdrは抽出子(extractor)と呼ばれ，a, b, nilは定数述語と呼ばれる．Lingのアルゴリズムでは，目標のプログラムが扱う任意のデータに対し，それらを一意に表現できるような抽出子と定数述語が(それらの定義も含めて)予め与えられることを仮定している．

論理トレースは，もとの正事実のみを被覆する，いわば，その事実と等価な節である．したがって，論理トレースを仮説に付加するだけでは，与えられた事実を丸暗記しているのと変わりはない．そこで，次の一般化則(1), (2)を繰り返し適用することによって仮説の一般化を行なう．

[10] たとえば，グランドアトムの真偽だけでなく，一般の節に関する真偽にも答えられなければならない．

(1) 仮説中の節 $P \leftarrow Q$ と定数述語 c に対し，$c(x_1)$ と $c(x_2)$ が Q に現れているとき，Q から $c(x_1)$ と $c(x_2)$ を取り去り，x_1 と x_2 を単一化することによって得られる節 $P' \leftarrow Q'$ をすべて仮説に付加する．

たとえば，$0(x)$ が定数述語のとき，節[11]

$$mul(x,y,z) \leftarrow 0(x), p(y,y_1), 0(y_1), 0(z).$$

が仮説中に存在する場合，以下の3つの節が新たに仮説に付加される．

$$mul(x,y,x) \leftarrow p(y,y_1), 0(y_1).$$
$$mul(x,y,z) \leftarrow p(y,x), p(z,z_1), 0(z_1).$$
$$mul(x,y,z) \leftarrow 0(x), p(y,z).$$

この操作は，3.2節で与えた精密化子における2の操作に対応している．ただし，同じ定数述語に現れているという制約がついている分，こちらの方がより構成的であるといえる．

(2) 仮説中の2つの節 $A \leftarrow B, C \leftarrow D$ において B の部分集合 $B1$ が D のインスタンスとなっている場合，すなわち，ある代入 θ が存在して，$B1 = D\theta$ であるとき，節 $A \leftarrow B$ の $B1$ の部分を $C\theta$ で置き換えることによって得られる節 $A \leftarrow B'$ を仮説に付加する．

たとえば，仮説中に

$$add(x,y,z) \leftarrow 0(x), p(y,y_1), 0(y_1), p(z,z_1), 0(z_1).$$
$$add(x,y,z) \leftarrow 0(x), 0(y), 0(z).$$

が存在する場合，

$$add(x,y,z) \leftarrow p(y,y_1), p(z,z_1), add(x,y_1,z_1).$$

のような再帰的な節が生成される．

有限個の論理トレースに対して，(1), (2) の一般化則は有限回しか適用できないので，それらの適用の繰り返しは有限時間内に停止する．こうして正事実だけを基に得られた仮説は，不適切に一般化された節を含んでいる場合があるから，この後，負事実との無矛盾性のテストが行なわれ，矛盾が起きた場合には，Shapiro の矛盾点追跡アルゴリズムを使って不適切な節の削除が行なわれる．Summers の THESYS とのもっとも大きな相違点は，この負事実と矛盾点追跡アルゴリズムの利用にある．後ろに矛盾点追跡アルゴリズムが控えていてくれるおかげで，THESYS と比べて網羅的な一般化が行なえるのである．

このアルゴリズムによって，抽出子と定数述語だけから定義できる論理プログラムのクラスを極限において同定できる．また，このアルゴリズムの自然な拡張として，Banerji の理論名辞生成法を実現できる．ただ，その場合のアルゴリズムの収束性(仮説の収束性)等に関する結果は得られていない．

[11] $p(x,y)$ は整数データに対する抽出子であり，$y = x - 1$ を表す．

4.3 文法学習における非終端記号の生成

文脈自由文法(CFG)は確定節文法(DCG)によって論理プログラムとして表現できる．たとえば，言語

$$L = \{a^n b^n \mid n \geq 1\}$$

を生成する文法

$$G = (\{S, A, B, C\}, \{a, b\}, P, S)$$
$$P = \left\{ \begin{array}{l} S \rightarrow aA, A \rightarrow aB, A \rightarrow b, \\ B \rightarrow aBC, B \rightarrow bC, C \rightarrow b \end{array} \right\}$$

は次のような論理プログラムとして表現できる．

$$S([a|x], y) \leftarrow A(x, y).$$
$$A([a|x], y) \leftarrow B(x, y).$$
$$A([b|x], x).$$
$$B([a|x], y) \leftarrow B(x, z), C(z, y).$$
$$B([b|x], y) \leftarrow C(x, y).$$
$$C([b|x], x).$$

$N(x,y)$ は差分リスト $x - y$[12]で表される文字列が N に対応する非終端記号から生成されるときに真となるような述語である．

このような対応関係の下で，言語 L を生成するような文法 G を見つけるという文法学習の問題は，$S(\_,\_)$ に関するモデルだけを与えた状態で，$S(\_,\_)$ 以外の理論名辞 $A(\_,\_), B(\_,\_), \ldots$ を生成しつつ上のようなプログラムを見つけるという拡張されたモデル推論問題として捉えることができる．すなわち，モデル推論における理論名辞の生成問題は，文法学習における非終端記号の生成問題に対応する．この例にあるプログラムからも分かるように，一般の論理プログラムと異なり，必要な述語の形(引き数の数)や節の構造が予め固定されているので，効果的な理論名辞(非終端記号)を効率的に生成できることが予想される．

石坂は正則言語[7]や単純決定性言語[8]のクラス[13]に対する文法学習において，必要な非終端記号の生成が効率的に行なえることを示すことによって，それらのクラスが多項式時間学習可能であることを示している．その中でもっとも重要な点は，各非終端記号がそれぞれ適切なモデルをもった形で生成されるということである．モデルをもった非終端記号が生成されることによって第2節で述べたような矛盾点追跡アルゴリズムの効果を反映でき，適切なモデルをもっていることによって正しい文法に収束することが保証される．横森[19, 20]は単純決定性言語のもつ性質を一般化することによって，非終端記号生成が可能な新たな言語のクラスを導入し，そのクラスの多項式時間学習可能性に関する結果を導いている．

[12] たとえば，$[a,b,b,a] - [b,a]$ はリスト $[a,b]$ を表す．

[13] 例として挙げた言語 $\{a^n b^n \mid n \geq 1\}$ はこのクラスに属する．

5 おわりに

以上，Shapiroによるモデル推論と，極く一部ではあるが，それに関連した最近の研究について紹介してきた．本稿では，特に，理論名辞の問題に焦点を当てたが，この他にも，Lairdによる精密化の概念の拡張[9]や，石坂による節の頭部の生成における最小汎化利用の試み[6]等も行なわれている．また，モデル推論の応用として，有川等によるEFS(elementary formal systems)を表現系とした言語学習への応用[3]や，劉による制約付き再帰的図形の学習への応用[22]等が試みられている．本稿では紙面の関係で十分な解説ができなかったところが多いので，興味ある読者は参考文献を参照して頂きたい．

参考文献

[1] D. Angluin. Learning k-bounded context-free grammars. Research Report 557, Yale University Computer Science Dept., 1987.

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[3] S. Arikawa, T. Shinohara, and A. Yamamoto. Elementary formal system as a unifying framework for language learning. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pp. 312–327. Morgan Kaufmann, 1989.

[4] R. B. Banerji. Learning theories in a subset of a polyadic logic. In *Proc. Computational Learning Theory '88*, pp. 281–295, 1988.

[5] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[6] H. Ishizaka. Model inference incorporating generalization. *Journal of Information Processing*, 11(3):206–211, 1988.

[7] H. Ishizaka. Inductive inference of regular languages based on model inference. *International journal of Computer Mathematics*, 27:67–83, 1989.

[8] H. Ishizaka. Polynomial time learnability of simple deterministic languages. *Machine Learning*, 5(2):151–164, 1990.

[9] P. D. Laird. *Learning from Good and Bad Data*. Kluwer Academic, 1988.

[10] X. Ling. Inventing theoretical terms in inductive learning of functions - search and constructive methods. In Z. W. Ras, editor, *Methodologies for Intelligent Systems, 4*, pp. 332–341. North-Holland, October 1989.

[11] X. Ling. Learning and invention of horn clause theories - a constructive method. In Z. W. Ras, editor, *Methodologies for Intelligent Systems, 4*, pp. 323–331. North-Holland, October 1989.

[12] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1984.

[13] S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proc. 5th International Conference on Machine Learning*, pp. 339-352, 1988.

[14] S. Muggleton and W. Buntine. Towards constructive induction in first-order predicate calculus. TIRM 88-031, The Turing Institute, 1988.

[15] G. D. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pp. 153-163. Edinburgh University Press, 1970.

[16] E. Y. Shapiro. Inductive inference of theories from facts. Technical Report 192, Yale University Computer Science Dept., 1981. (有川節夫訳：知識の帰納推論，共立出版，1986).

[17] E. Y. Shapiro. *Algorithmic program debugging*. PhD thesis, Yale University Computer Science Dept., 1982. Published by MIT Press, 1983.

[18] P. D. Summers. A methodology for lisp program construction from examples. *JACM*, 24:161–175, 1977.

[19] T. Yokomori. Learning context-free languages revisited. In *Proc. of the Japanese-Czechoslovak Seminar on Theoretical Foundations of Knowledge Information Processing*, pp. 93–102, June 1989.

[20] T. Yokomori. On learning a class of context-free languages in polynomial time. In 数理解析研究所講究録 *716*:短期共同研究「計算量理論 とその周辺」, pp. 60–70. 京都大学数理解析研究所, 1990.

[21] 石坂裕毅. 述語生成に関する一考察. Thechnical Memorandum TM-0631, ICOT, 1988.

[22] 劉樹岑. モデル推論による制約付き再帰的図形の学習. In ソフトウェア基礎論研究会報告 *No.34*. 情報処理学会, 1990.