最汎アトムを用いない精密化方法による Prolog プログラムの帰納的自動合成システムの，C 言語による実現

鈴 木 昇 一，中 村 三 郎

A Technique for Prolog Program Synthesis by Doing without Most General Atoms

Shoich SUZUKI・Saburo NAKAMURA

E.Y. Shapiro proved that any Prolog-program is constructed by means of an iteration of the following four operations:
(1) An addition of an atom to a head of a Horn clause
(2) A unification of two variables (3) A unification of a variable and a function (4) An addition of the most general agom to a body.
It is indeterminate to obtain the most general atom needed in the operation (4). Therefore in this investigation the inductive model-inference algorithm proposed by Shapiro has been implemented with C-language without using the most general atom. It is easy to control the program-synthesis in this method, but a space of hypotheses increases explosively in the size. We shall show that the contradiction-backtracing is reduced to the inverse application–order of the resolution principle.

1．まえがき

仕様（program specification）からプログラムを合成するには，仕様を次の3つのいずれかで与えれればよい： 自然言語，形式論理，例題提示。 △

帰納の働きが人間の思考の段階で発現することにより，各種各様の知識を利用・推論し，人間は仕様からプログラムを作っていると思われる。コンピュータにプログラム作りをさせるに必要な各種の知識は曖昧で不完全な形でしか与えられない。この事態は自然言語や形式論理を用いてプログラム仕様を表現してもそうである。どうせそうなら，この2つの仕様よりも一層不完全な仕様としての例題（合成すべきプログラムの入出力例）提示で，プログラムを合成する試みはそんなに無意味なことではなかろう。

特に，例題提示仕様によるプログラム合成の場面では，閉じた形の推論ではなく，推論の各段階で各種各様の知識を取り入れながら，仮説を生成し，仮説空間を探索し，得られている仮説集合を増加・減少させるという意味で開いた形の推論，いわゆる非単調推論の働きをコンピュータに与えなければならない。帰納を演繹的な推論で近似できるような計算モデルとしての「演繹（体系を用いた仮説）推論」の枠組が必要とされる。

論理型言語 Prolog で書かれた，n 引数 X_1, X_2…, X_n の論理プログラム

$P(x_1, x_2, \cdots x_n)$

を次の条件で推論・固定できるだろうか？：

t_1, t_2, …t_n を変数が含まれぬ n 個の項として，$P(t_1, t_2, \cdots t_n)$が真あるいは偽となるような n 組

$\langle t_1, t_2, \cdots t_n \rangle$

をその真偽と共に与える。　△

E.Y. Shapiro は合成されるべき論理プログラム P にある条件を課し，プログラムの合成を可能にするシステム MIS（Model Inference System）を，帰納推論メカニズムが計算モデル（演繹推論の枠組）で実現される形式で研究している。$P(t_1, \cdots t_n)$ が真，偽であるような n 組

$\langle t_1, \cdots, t_n \rangle$

を各々正，負の入力例といい，正の入力例 $\langle t_1, \cdots, t_n \rangle$ に対しては

$M(P(t_1, \cdots, t_n)) = \text{true}$

であり，また，負の入力例 $\langle t_1, \cdots, t_n \rangle$ に対しては

$M(P(t_1, \cdots, t_n)) = \text{false}$

であるような

1階言語 L のグランドアトム（ground atom；変数を含まない原子論理式）の集合から集合 {true, false} への写像 M

をモデル（model）という。（実は意味論からすれば，M から一階述語 $P(t_1, \cdots, t_n)$ の真偽が定まると考えなければならないが。）MIS では，この種のモデル M を極限において固定する（identify in the limit）とは，M で真なるすべてのグランドアトムを導出証明できるような，M で真なる仮説（確定節）の有限集合（論理プログラム）を帰納推論すること（M のアトム完全公理化；atomic-complete axiomatization）であるとみなし，

負の入力例を導出証明できたとき（推測がモデル M に関し強過ぎるとき），合成途中の論理プログラム（推測；conjecture）中の誤まった仮説（反駁（refutation）された仮説；矛盾点）を見い出し，推測から削除しその旨登録し，また，正の入力例が導出証明できないとき（推測が弱過ぎるとき），以前反駁された仮説を精密化（特殊化）していく戦略

が採用されている。

このため，MIS を実現するには，推論途中の推測としての理論（合成されるべき Prolog プログラム）が正，負の入力例を導出証明できるかどうかの判定のために，

(1) 導出原理アルゴリズム（resolution algorithm）

が必要とされ，また，推測中の誤まった仮説を見いだすために，

(2) 矛盾点追跡アルゴリズム（contradiction backtracing algorithm）

が必要とされ，さらに，反駁された仮説を精密化するために，

(3) 精密化アルゴリズム（refinement algorithm）

が必要とされ，

モデル推論アルゴリズム（model inference algorithm）

はこの3つのアルゴリズム(1)～(3)から成っている。

本研究は帰納的推論法としての，Shapiro のモデル推論アルゴリズムをほぼ忠実に C 言語で実現したものであって，事実から理論を帰納的推論（inductive inference of theories from facts）する際，格別原理的な改良を行なった訳でもないが，

(a)矛盾点追跡アルゴリズムを導出原理アルゴリズムに帰着させたこと

(b)任意の Prolog プログラムを生成できる簡便な精密化アルゴリズムをいわゆる最汎アトムを用いない形で実現したこと

が特色である。

2．導出原理と矛盾点追跡との関係

2つのアトム（原子論理式）G_i，Aから成る集合 $\{G_i, A\}$ の最汎単一化作用素（most general unifier; mgu）θ が存在して，

$$P \equiv G_i \cdot \theta = A \cdot \theta \qquad (2.1)$$

が成り立つならば†，目標節（goal clause）

$$G = [\,? - G_1, G_2, \cdots, G_{i-1}, G_i, G_{i+1}, \cdots - , G_\ell\,,\,] \qquad (2.2)$$

と確定節（definite clause）

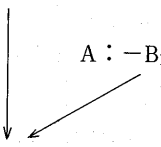
$$C = [A : - B_1, B_2, \cdots, B_m] \qquad (2.3)$$

との両者から††

$$G' = [\,? - (G_1, G_2, \cdots, G_{i-1}, B_1, B_2, \cdots, B_m, G_{i+1}, \cdots, G_e) \cdot \theta\,.] \qquad (2.4)$$

を導出してもよい，いいかえれば，

$? - G_1, \cdots, G_{i-1}, G_i, G_{i+1}, \cdots, G_\ell.$



$? - (G_1, \cdots, G_{i-1}, B_1, \cdots, B_m, G_{i+1}, \cdots, G_e) \cdot \theta.$

という図式が得られるというのが，ホーン節（Horn clause；確定節と目標節の総称）の有限集合における J.A. Robinson による導出原理（resolution principle）を適用した結果としての，SLD（Linear resolution with Selection function for Definite clause）導出である。

このとき，式（2.4）を2式（2.2），（2.3）に関する2分導出形（resolvent），SLD導出形といい，式（2.1）のPを導出に使われたアトム（atom resolved upon）という。また，G_i，Aを導出が行なわれるアトムという。導出形を作成する過程を導出証明（resolution）と呼び，式（2.2）を導出証明の左成分といい，式（2.3）を右成分という。また，式（2.3）のCを式（2.2）の目標節Gに対する入力節という。

現在のPrologシステムで多く用いられているのは常に i=1 とすることであり，いいかえれば，式（2.3）のAと，常に式（2.2）のGの一番左側のアトム G_1 とを導出が行なわれるアトムとすることである。また，現在のPrologシステムでは，式（2.4）の mgu θ において，変数の代りに置き換えられる項にその変数が登場してはならないという出現チェック（occur check）を完全にはしていない。さらに，プログラム節ともいわれる確定節の有限集合Sから式（2.3）の確定節Cを選んでくる順番はS内で並べられている順番であり，いわゆるSLD木を縦型に探索する。もし，出現チェックを完全に行なったとしても，縦型探索方法と，入力節の固定順序づけ（例えば，S内の順番）とによるSLD導出では，導出原理のもつ論理完全性[2]は保証されないことに注意しておく。

矛盾点追跡アルゴリズムは（モデルMで）偽のプログラム節を検出するため，導出原理の上述の適用方向を逆にたどる方向を指摘する。

まず，2式（2.2），（2.3）から式（2.4）を導出すること，つまり導出原理は導出に使われた，式（2.1）のアトムPが真であって

---

† 以後，DEC－10 Prologの記法を使用する。

†† 式(2.3)の確定節Cの頭部，体部とは各々，A，$\{B_1, B_2, \cdots, B_m\}$ のことであり，このCは論理的には

if B_1 and B_2 and … and B_m
then A

の意味を持っている。確定節においては頭部に出現する変数記号は必らず体部に出現しているのが普通である。また，体部が空の確定節Cは

A：－.

と書かれ，これは

if 恒真 then A
［無条件にAが成立する）

の意である。また，式(2.2)の目標節Gは頭部がない確定節と考えられ，次の意である：

if G_1 and G_2 and … and G_l
then 恒偽
（$G_1, G_2, \cdots, G_l$ が共に成り立つのは矛盾である）

も偽であっても，いいかえれば，Pの真偽が不明であっても適用できることに注意する。導出原理は，式（2.1）の，導出に使われたアトム

$P=G_i\cdot\theta=A\cdot\theta$

が真の場合，式（2.3）のプログラム節Cはおのずから真となるので，式（2.2）の目標節Gから，

$?-(G_1, G_2, \cdots, G_{i-1}, G_{i+1}, \cdots, G_e)\cdot\theta$ (2.5)

が得られ，また，Pが偽の場合，式（2.2）のGはおのずから真となるので，式（2.3）のCから

$?-(B_1, B_2, \cdots, B_m)\cdot\theta$ (2.6)

が得られることを説明しており†，この事実に注目し，

式（2.1）の，導出に使われたアトム

$P=G_i\cdot\theta=A\cdot\theta$

が真であれば，式（2.4）のG′から偽になる可能性をもつ式（2.5）つまり式（2.2）のGへとたどり，また，Pが偽であれば，式（2.4）のG′から偽になる可能性をもつ式（2.6）つまり式（2.3）のCへとたどればよい

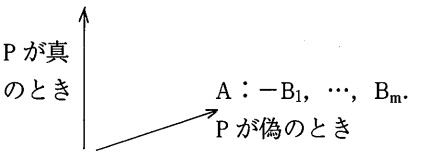
ことを指摘するのが矛盾点追跡アルゴリズムであり，式（2.3）のCにたどりついたとき，矛盾点つまり偽になるプログラム節Cが発見されたことになる。

要約していえば，矛盾点追跡アルゴリズムは，導出原理を適用した際の，導出に使われたアトム

$P=G_i\cdot\theta=A\cdot\theta$

が真あるいは偽に応じて

$?-G_1, \cdots, G_{i-1}, G_i, G_{i+1}, \cdots, G_\ell.$



$?-(G_1, \cdots, G_{i-1}, B_1, \cdots, B_m, G_{i+1}, \cdots, G_\ell)\cdot\theta$

というように，図示される二つの方向のいずれか一方へとたどればよいことを指摘している。

3．精密化

not述語とカットオペレータとを含まぬPrologプログラムは，次の二つのいずれかの形式(プログラム節)が混在した集まりである：事実節，規則節を一般に確定節というが，

A：−.　（事実節）

A：$-B_1, B_2, \cdots, B_m$.（規則節）

ここに，A，B_jは多引数（有限個）をもつ原子論理式（atomic formula；論理記号を含まぬ論理式）である　△

従って，Prologプログラムは次の4つの精密化操作 RFO_1〜RFO_4 の有限回の適用で得られることになる。

今，Reynoldsの定義を採用して，確定節Eあるいは関数fの大きさ size(E)，size(f)を次のごとく決める：

size(E)＝[Eに含まれる（区切り記号を除いた）記号の出現回数からEに含まれる相異なる変数の個数を差し引いたもの]††

size(f)＝[size(E)の定義において，Eの代りにfを採用した定義]†　△

また，size aの確定節Aからsize bの確定節Bを作ったとき，そのサイズの変化　△ size(B，A）を次の様に定義する：

---

† これが式(2.4)の表現G′が成り立つ理由である。

†† 述語名，変数名，関数名は2個以上の文字から成る一つの文字列で構成されている場合も一つの記号と考える。

△size(B, A)=b−a. △

(i) 精密化操作 RFO_1

(頭部へのアトムの追加；空節からサイズ1のアトムを頭部にもつ確定節を作る)

A=(？−.) (空節)

から

$B=(E(x_1, x_2, \cdots, x_n) : -.)$

を作る。ここに,

size(A)=0

$size(B)=(n+1)-n=1$

であり,

△size(B, A)=1−0=1 △

以下のii, iii, ivでは次の定義を用いる：一階述語論理でのある言語Lを選定する。Lの文としての確定節Aを

A=(E：−F.)

とする。Eは空または一つのアトム，Fは空またはアトムの有限列 $F_1, F_2, \cdots$ である。ここに，E, F_j は言語Lから任意に選んだ一つのアトムである。 △

(ii) 精密化操作 RFO_2

(2つの変数の単一化)

Aに出現する変数のうち，2つの相異なる変数を x_i, x_j として，A内の x_i の出現箇所のすべての出現箇所を x_j でおきかえたものを B とする。k, ℓ を

k：Aでの，記号の出現回数

ℓ ：Aでの，相異なる変数の個数

とすると，sizeの定義により

$size(A)=k-\ell$

である。そうすると,

$size(B)=k-(\ell-1)$

$=size(A)+1$

∴ △size(B, A)=1

(iii) 精密化操作 RFO_3

(変数と関数との単一化)

Aに出現する変数のうち，唯一つの変数 x_i に注目し，A内の x_i の出現箇所のすべての出現箇所を,言語Lから任意に選んだ関数

$f(Y_1, Y_2, \cdots, Y_m)$

でおきかえたものを B とする。ここに，Y_1, Y_2, …, Y_m はAに出現しない相異なる変数である。

A内で x_i の出現箇所がk個あるものとする。

$size(f(Y_1, \cdots, Y_m))$

$=(m+1)-m=1$

であり,

$size(A)=n-\ell$

とすると,

Bに含まれる記号の出現回数

$=(n-k)+k(m+1)$

Bに含まれる相異なる変数の個数

$=(\ell-1)+m$

であるから,

size(B)

$=(n-k)+k(m+1)$

$-[(\ell-1)+m]$

$=n+km-\ell+1-m$

$=size(A)+m(k-1)+1$

∴ △size(B, A)$=m(k-1)+1$

(iv) 精密化操作† RFO_4

(体部への，アトムの追加)

Aの体部に言語Lから任意に選んだ一つのアトム

$G(z_1, z_2, \cdots, z_n)$

を付加したものを B とする††：

---

† 精密化操作 RFO_1 は精密化操作 RFO_4 の特別な場合である。

†† Shapiroの提案している精密化方法と異なる点は,Gが B=(E：−F, G.)が既約であるような，A=(E：−F.)に関する最汎アトム(後述)とは限らないことである。このことが合成されるPrologプログラム(MISの出力としての推測)には多くのゴミ(その論理的能力を増加させない余分な仮説としての確定節)が存在する原因の一つとなる。

B=(E：−F, G.)

ここに，z_1, z_2…, z_n は E, F に出現しない変数の集まりである。

size(G)=(n+1)−n=1

であり，

size(A)=m− ℓ

とすると，

size(B)=(m+n+1)−(ℓ+n)

=m− ℓ +1

=size(A)+1

∴ △size(B, A)=1 △

上記の4つの精密化操作 RFO_1〜RFO_4 の内，RFO_1, REO_2, REO_4の適用においては，サイズが丁度1だけ増加するが，RFO_3の適用では，サイズが

m(k−1)+1

だけふえる。この増加は，A内でm引数の関数 $f(Y_1, \cdots, Y_m)$ で置きかえようとする変数 x_i の出現箇所（この箇所が丁度k個ある）が多いほど，また，引数の多い関数で置き換えるほど，大である。もし，

k=1

であれば，その増加 △size(B, A) は

△size(B, A)=1

となるが。

（例3.1）X, Yを非負整数として，2引数述語 le(X, Y) を，

X≦Y のとき真，X>Y のとき偽を返す述語

と定義すれば，関数 S(X) を

S(X)：Xの後続要素(=X+1)を返す関数

として，次の2行から成るプログラムが le(X, Y) である：

le(X, X)：−.

le(X, S(Y))：−le(X, Y). △

このプログラムは次のFig.3.1のような精密化過程で得られる。本図において，枝に付けられた番号は精密化操作 (i)〜(iv) のいずれを適用したかを示している。

4. MISでの帰納的推論アルゴリズム

合成途中のPrologプログラム（プログラム節の集合）から成る仮説空間は，

(i) モデルMにおいて偽なる観測文[2]を導出するとき，モデルMに関して強過ぎるといい，

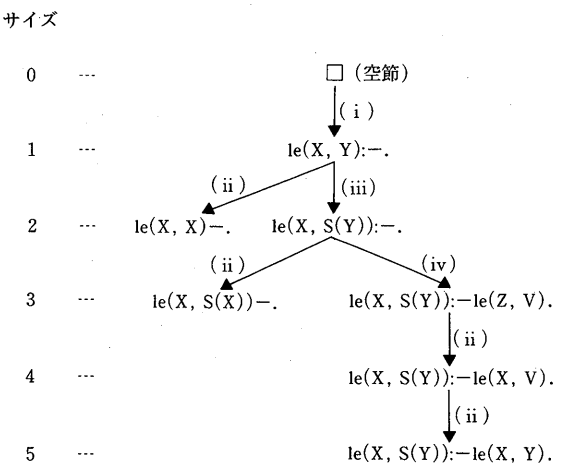


Fig. 3.1 le(X, Y) を得る精密化過程

(ii) モデルMにおいて真なる観測文を導出できないとき，弱過ぎる

という。

観測文とは一階述語論理言語Lでのグランドアトムのことであり，仮説空間内の点とは確定節（プログラム節ともいう）のことであるが，現在得られている推測（理論）としての仮説空間がモデルMに関して強過ぎるとき，少なくとも仮説空間内の一つの仮説が偽であると結論でき，この仮説（矛盾点）を検出するのが矛盾点追跡アルゴリズムである。MISの基本動作は，真の例としての観測文の集合を導出証明するプログラム節（仮説）を，

偽の例としての観測文を導出証明する矛盾点としてのプログラム節を排除しこの排除されたプログラム約を精密化すること

で得ようとするものである，と説明される。

さて，モデルMに関し，Mに問い合わせた結果，その真偽が判明したグランドアトムのことを オラクル（oracle，神託）という。矛盾点追跡アルゴリズムにおいて，既知のオラクル集合が矛盾点の追跡に必要なオラクルを含まない場合があるが，その対策として，

「オラクルの不足分はそれ迄に生成したモデルの実行結果で補うことなしに，要求すれば得られる」

という立場を採用し，

「オラクルからの返答はすべて貯えておいて二度目以降はこの返答を利用する」

ことにする。

以下に述べる帰納的推論アルゴリズムを，C言語で書かれたCprologに，矛盾点追跡アルゴリズム，精密化アルゴリズムを付加し，作製した。

現在の仮説空間（推測としてのPrologプログラム）をTとして，T内の文（確定節）のサイズの最大値をkとする。2つの集合S_f，S_tには現在迄に読み込んだ偽，真の観測文を各々，貯えておく。各時刻に一つずつ観測文を読み込むことにして，時刻nに読み込まれた観測文α_nとその真理値V_nの対$F_n=\langle \alpha_n, V_n\rangle$を事実と呼んでおく。

帰納的推論アルゴリズム

0．初期化

k：=0，S_f=｛空文｝

S_t：=｛　｝（空集合），T′：=｛空文｝，

n：=1

手順

1st repeat

1．事実$F_n=\langle \alpha_n, V_n\rangle$を読み，$V_n$がtrueあるいはfalseに従い，$\alpha_n$を$S_f$あるいは$S_t$に追加する。

2nd repeat

2．（Tが強過ぎる場合）

1st while（ある$\alpha \in S_f$がnステップ以内でTから導出証明される）

do

3．矛盾点追跡アルゴリズムを適用し，T内の偽の仮説（矛盾点）をTから除去し，“false”とマークする。

4．（Tが弱過ぎる場合）

2nd while（ある$\alpha_i \in S_t$がh(i)ステップ以内でTから導出証明されない）

do

5．上記の3において，“false”とマークされたすべての仮説を，精密化アルゴリズムを用いて，現在のT内の文のサイズの最大値kを1つだけ増加させる範囲内で精密化し，Tにこの様にして得られた精密化文の全体を付加したものを改めてTとする。

6．2nd until

推測Tがこれまでに読み込まれた事実F_1，F_2，…，F_nに関して強過ぎることもなく，弱過ぎることもない。

7．その論理的推論能力を増加させない多

くの余分な仮説をTから除去した後（推測Tの局所的最適化），Tを出力する。

8．1st until

n：＝n＋1

△

C言語で上述のアルゴリズムを表現する際，以下の2点Ⅰ，Ⅱのごとく配慮した。

Ⅰ．上述の段階4におけるhは全域的な帰納的関数でなければならない。本研究では，mを十分大きくとり，h(i)を次の様に選んだ。

h(i)＝i if i≦m，＝m if i＞m.

Ⅱ．実際は，段階4，5において，精密化操作を1回適用する度に2nd whileの繰り返し条件の成立・不成立を確かめ，成立すれば（導出証明されなければ）その度に矛盾点追跡アルゴリズムを呼び稼働させる。

その際，次のi～ivを実行する。

(i) 精密化を終えた仮説を除去し，その仮説に“fin”というマークをつけ，仮説空間内で仮説の番号を付け換え，ゴミの掃除をする。

(ii) 精密化して得られた仮説に，この仮説が“true”か“false”かが判明していないという意味で，“idle”というマークをつける。

(iii) これ迄に読み込まれた観測文の内，偽の観測文を導出証明するのに必要とされた仮説の内矛盾点（偽の仮説）に“false”というマークをつける。

(iv) これ迄に読み込まれた観測文の内真の観測文を導出証明するのに必要とされた仮説に“true”というマークをつける。

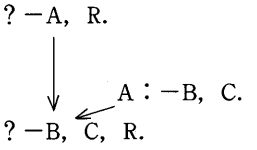
以上のi～ivにより，推測から一つの仮説を“false”とマーク付けして除去することがそれ以前には仮説から導出可能であった，S_t内のある観測文を導出不能にするかどうかをテストできることになった。

5．矛盾点追跡アルゴリズムの実現

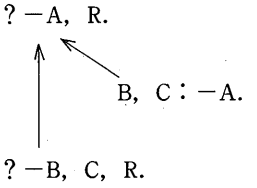
空節導出（導出原理）アルゴリズムとは独立に，矛盾点追跡アルゴリズムを作製し，MISを稼働させたところ，ずいぶん所要時間が大なので，ホーン節（Horn clause；確定節，目標節の総称）のhead，bodyを入れ換える操作を導入し，空節導出アルゴリズムに帰着させたところ，以前として比較し，

所要時間が約1/3

になった。head，bodyを入れ換えるこの原理は，簡単にいえば，



という導出過程の，逆にたどる矛盾点追跡過程を



という導出と考えることで説明される。

矛盾点追跡過程において配慮したことは次の3点である。

(i) ホーン節が真か偽に関し神託が返してきた情報は登録しておいて利用する。

(ii) 変数が残存した場合は言語Lの最も簡単な定数を代入する。

(iii) 仮説空間（推測，理論）Tが

le(X，S(Z))：－le(X，Y)，le(Y，S(Z))　(5.1)

le(S(X)，Y)：－le(S(X)，X).　(5.2)

の場合で考えよう（3章の例3.1を参照）。

このTから

？－le(S(S(0))，S(0)).　(5.3))

が導出証明され得るかどうか（第4章での帰納的推論アルゴリズムにおいて1st whileの繰り返し条件が成立するかどうか）の検証を

考えよう。

以下の説明は，矛盾点を矛盾点追跡アルゴリズムの稼働なしに発見する方法の一つで，確定節の head，body に同一グランドアトムが生じるかどうかで，矛盾点を発見するものである。

2式(5.1)，(5.3)を比較する。X に S(S(0)) を，また，Z に 0 を代入すると，式(5.1)は

$$le(S(S(0)),\ S(0)) : -le(S(S(0)),\ Y),\ le(Y,\ S(0)). \qquad (5.4)$$

と書き直される。また，2式(5.2)，(5.3)を比較する。X に S(0)を代入して，式(5.2)から

$$le(S(S(0)),\ Y) : -le(S(S(0)),\ S(0)). \qquad (5.5)$$

を得る。式(5.4)に式(5.5)を代入すると，

$$le(S(S(0)),\ S(0)) : -le(S(S(0)),\ S(0)),\ le(Y,\ S(0)). \qquad (5.6)$$

が得られるが，head，body に同一グランドアトム

$le(S(S(0)),\ S(0))$

が生じていることがわかる。この際，式(5.2)で表わされる仮説を矛盾点追跡アルゴリズムで発見した矛盾点と同様な扱いとした。

6．仮説空間上の半順序

仮説空間の2つの元の間には，

$\leqq_\rho$

という半順序（partial ordering）がついている。ここに，ρ は仮説言語（仮説を表現するための言語）L_h の文に対する

精密化演算子（refinement operator）であり，

A，B $\in L_h$ として $A \leqq_\rho B$ での $\leqq_\rho$ は次のi，ii，iiiを満たす：

(i) 反射律（reflexive law） $A \leqq_\rho A$

(ii) 推移律（transitive law）

$A \leqq_\rho B$ かつ $B \leqq_\rho C$ ならば $A \leqq_\rho C$

(iii) 反対称律（antisymmetric law）

$A \leqq_\rho B$ かつ $B \leqq_\rho A$ ならば $A = B$ △

上述を更に説明しよう。包含関係

$L_h \leqq L$

を満たす一階言語 L を導入する。A，B∈L に関し，2条件

(a) A がB を導出証明する

(b) size(A)≦size(B)

が成立するとき，

B はA の精密化（refinement)である

という。このとき

$B \in \rho(A)$

と書く。ここに，

$\rho(A) \equiv \{B \mid A \in L_h$ に精密化演算子 ρ を適用すると，$B \in L_h$ が得られる$\}$

半順序関係 $\leqq_\rho$ の定義は次の通りである：

$A \leqq_\rho B$ が成立するとは，

$A_0 = A$，$A_n = B$ であるような

$A_j \in \rho(A_{j-1})$，$j=1, 2, \cdots, n$ が存在することをいう。

以上から，$A \leqq_\rho B$ が成立しているとき，言語 L のあるモデル M において，

(i°) A が真であれば，B は真であるが成り立ち，このi°の対偶として，

(ii°) B が偽であれば，A は偽であるが成り立つ。この性質ii°が論理的能力を増加させない多くの余分な仮説を除去するのに使われる。注意すべきは，i°からわかるように

(iii°) A が偽であっても，B が真になる可能性はあるけれども

(iv°) A が真であれば，B は偽になる可能性はないことである。iii°，iv°は以下の理由で第3章での精密化操作を実施することが無意味でないことを説明している：　精密化操作の内任意の一つの操作の適用によって，確定節 A から確定節 B が得られたとき，

$A\theta = B$

となる代入θが存在し，従って，

Aが B を導出証明し，

一方，

size(A)＜size(B)

なので，

B はA の精密化となっている。

7．最汎アトム

アトム F_i の集合

$F=\{F_1, F_2, \cdots\}$

に関し，

E：−F.

は

E：−F, F_2, ….

の意であり，この確定節は，アトムの集合

{E, F_1, F_2…}

と同一視される。節形式では，～を否定記号として，

{E, ～F_1, ～F_2, …}

と同一視されるものであるが。

上述の約束の下で，

アトムの集合 S の既約性

から説明する。アトムの集合 S は，次の条件が成立すれば，既約（reduced）であるという：

$S\cdot\sigma \subsetneq S$ なる代入σが存在しない　△

第 3 章での精密化操作 RFO_4 に関連していえることは，付加されるアトム

$G=G(Z_1, Z_2, \cdots, Z_n)$

は

E：−F, G.

が既約であるような，

E：−F.

に関する最汎アトム（the most general atom）であることが望ましい。

この望ましい理由は，最汎アトムに関する次の説明から理解できる：

アトム $G=G(Z_1, \cdots, Z_n)$ における変数 $Z_1, \cdots, Z_n$ が E，F に出現してもよい変数の集まりとする。

(i)　E：−F. が既約であること

(ii)　2 条件

$Q\cdot\theta=G$ かつ $(E:-F.)\theta=(E:-F.)$ が成立する代入θが存在する任意のアトム Q（このような代入θが存在する 2 条件を満たすアトム Q は E：−F. に関し G より一般的であるという）に対し，

E：−F, Q.

が既約でないこと。

以上の 2 条件 i，ii が成立するとき，G は E：−F, G. が既約であるような，E：−F. に関する最汎アトムであるという。

なお，精密化操作 RFO_4 の直後に RFO_2, RFO_3 を適当な回数適用することは，RFO_4 において体部に付加されるアトムを最汎アトムにする効果をもたらすことに注意しておく。

8．帰納推論型プログラム合成システムの構成

MIS を帰納推論型プログラム合成システムとして構成したが，その主要モジュールは次の 6 つである：

main：　処理系制御モジュール（C 言語で，135ステップ）

hinp：　コマンド処理モジュール（741ステップ）

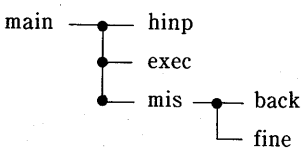
exec：　導出原理ロジックモジュール（735ステップ）

mis：　モデル推論システム MIS の本体（481ステップ）

back：　矛盾点追跡モジュール（278ステップ）

fine：　精密化モジュール（687ステップ）　△

三つのモジュール mis，back，fine をあわせると，1446ステップであり，これら 6 つのモジュール間の包含結合関係は次のように表わされる。



モジュール fine を起動すると，精密化のプロセスの様子が見られ，また，これ迄に貯わえられているオラクルからの返答も仮説空間も見ることができる。更に，精密化の途中でもし必要ならば言語 L を補強できる。

モジュール back を起動すると，空節への導出が何故うまくなされ得ないかも理解できる理由が提供される。

この MIS 開発の様相は次の通りである。

使用言語： C 言語

期間工数： prolog インタプリタの作成には1.3人月

MIS の作成（システム設計から総合テストまで）には 3.1人月

マシン： IBM5560k0l（640KB，プロセッサはクロック周波数8MHZ の80286）

プログラムステップ数： 5756ステップ

性能： 合成されるプログラムの確定節のサイズの最大値が 6 ～ 7 位。

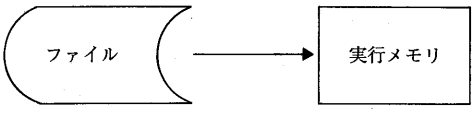
確定節のサイズの最大値が 5 であるプログラム

$$le(X,\ Y)=\begin{cases}\text{true if } X\leqq Y\\ \text{false if } X>Y.\end{cases}$$

を 6 個の事実例から 7 回の質問で 2 分位の所要時間で合成終了.

9．処理系 MIS の，データ構造の面から見た仕様

ホーン節，言語 L，事実例などは外部ファイルに格納されており，これを実行メモリ（導出に使われるホーン節を置くエリア）に移すことで，MIS が起動する。



ホーン節，言語 L，事実

また，次の 6 つのデータ構造が設けられている。（これら 6 つのデータ構造の間の関係は矢印等で結ぶことが出来る程簡単でない。）

カレントゴール	言語メモリ
ゴールヒストリ	事実メモリ
ホーンヒストリ	仮説メモリ

ゴールヒストリは導出の際，得られるゴール節の履歴をおくエリアである。カレントゴールは現在単一化（コニィファイ）中のゴール節（目標節）を置くエリアである。言語メモリは言語 L，組込み述語を置くエリアである。事実メモリは MIS の稼働前と稼働中にオラクルから得た知識と，稼働後更に要求して得られる知識とを貯わえておくエリアである。仮説メモリは主として，精密化操作により生成された仮説をおくエリアである。ホーンヒストリは矛盾点追跡を可能にするためにユニィファイ後のホーン節を貯わえておくエリアである。

この 6 つのデータ構造を使って，unify（単一化），backtrace（矛盾点追跡），refine（精密化）がなされる。unify，backtrace の動作を以下に具体的に説明する。

〈unify〉

(1) 仮説メモリ中の，偽でない仮説を抽出して実行メモリに置く。

(2) ゴールヒストリ中の現在のゴールを取出して，カレントゴールに置く。

(3) ユニィファイ可能なホーン節を実行メモリから選択する。

(4) カレントゴールとユニィファイ可能なホーン節とでユニファイを実行する。

(5) カレントゴールを取出して，ゴールヒストリ中の現在のゴールの位置に収める。

(6) ユニィファイ後のホーン節をホーンヒストリに収める。

(7) ユニィファイ回数を1だけ増加させる。

〈backtrace〉

(1) ゴールヒストリ中の，現在のゴールを取出してカレントゴールに置く。

(2) 対応するユニィファイ後のホーン節をホーンヒストリから取出す。

(3) カレントゴールとユニィファイ後のホーン節とで，第5章で説明したごとく，逆ユニィファイを実行する。

(4) カレントゴールを取出して，ゴールヒストリ中の現在のゴールの位置に収める。

(5) ユニィファイ回数を1だけ減少させる。

10. 確定節のとり得る6つの真理値

MISの稼働中仮説の集合としての合成途中のプログラムを構成する各確定節は

hold, null, idle, fin, true, false

の6つの真理値のいずれか一つを持つ。

言語Lで用いられる組込み述語は真理値holdをもつ。MISが稼働し始めるとき当初存在する確定節はすべてnullという真理値をもち，この確定節が精密化されて得られる確定節は精密化の直後はidleという真理値をもつ。真理値idleをもった確定節が無意味であると判明し合成プログラムから除かれると，finという真理値が与えられる。真理値idleをもつ確定節が矛盾点追跡アルゴリズムによって矛盾点と固定されたり，また，精密化操作RFO_4により生成された場合はfalseという真理値をもつ。真理値idleをもつ確定節がこれ迄に読み込まれた任意の真の観測文を導出証明することが判明したとき（帰納的推論アルゴリズムでの2nd whileを脱出したとき），trueという真理値が与えられる。

最後に，真理値trueをもつ確定節（事実節，規則節）の内規則節はそれ迄に読み込まれた真なる観測文を導出証明するかどうかを検証するとき（2nd whileに入るとき），真理値idleを持つように強制的に設定される。ただし，変数をもたない事実節に対しては，trueからidleへの，真理値の変更はしない。

11. 精密化の実施方法

4つの精密化操作

RFO_1：頭部へのアトムの追加

RFO_2：2つの変数の単一化

RFO_3：変数と関数との単一化

RFO_4：体部へのアトムの追加

については第3章で説明されたが†，帰納的推論アルゴリズムでの段階5での具体的な精密化の方法は次のように実施される。

1°. 精密化の実施については，操作RFO_1をまず行ない，その後，操作RFO_4，RFO_2，RFO_3の順に多く適用する。このとき，操作RFO_3をこれ以上適用しても論理的推論能力が増加しないという限界を早急に見つけるために，操作RFO_1を実行した後は頭部の変数に関し操作RFO_3をいたずらに適用しない。何故ならば，このようにしないと，

```
le(X, X) :- .
le(X, S(X)) :- .
le(X, S(S(X)) :- .
      ⋮
```

という様な無限列を得てしまうからである。いいかえれば，可能な限り，bodyのある確定節を仮説として採用する。

---

† 与えられた正，負の事実の集合に矛盾がないという条件の下では，この4種類の精密化を繰り返し適用していけば，与えられた任意の負の事実を説明しないように，そして，最後には，与えられた正の事実だけを説明する“事実節だけから成るPrologプログラム”まで精密化されることがある。

2°. 最汎アトムを用いないため，RFO_4を実行した後，直ちにRFO_3あるいはRFO_2を実行し，RFO_4-RFO_3，あるいはRFO_4-RFO_2をあたかも一つの操作RFO_4のように取り扱う。

なお，精密化操作RFO_3の適用は推測T内の節のサイズの最大値を2以上増加させる場合があるので，この場合はRFO_3の適用を延期しなければならない（帰納的推論アルゴリズムでの段階4，5を参照）。精密化し過ぎにより所要のプログラムが合成不能という事態の到来を阻止するためである。

3°. 仮説空間Tの各元（各仮説）にある一つの精密化操作を適用する際，その順序は次の通りである：

サイズの小さい順にまず行ない，次に，頭部のサイズの小さい順に，次に，確定節に含まれるアトムの数の小さい順に行なう。

4°. "false"とマークされる矛盾点を，矛盾点追跡アルゴリズムを適用しないで発見する方法として，次の4.1~4.3を実行する。

4.1 操作RFO_4を実行した後に直ちにこの仮説を"false"とマークする。

4.2 確定節

A(X, Y)：−B(X, V), C(W, Y).

の如く，体部のアトムに独立に登場した変数V，Wがある場合，この確定節（仮説）を直ちに"false"とマークする。

4.3 導出証明の段階で，目標節とユニィファイするある確定節のhead，bodyに同一グランドアトムを生じさせる確定節を"false"とマークする。

5°. 操作RFO_2を行ないその直後RFO_3を行なっても，また，RFO_3の実施直後RFO_2を行なっても，変数名の付け換えで同一になる確定節（仮説）の集合が出現するので，この集合を直ちに一つの仮説に表現し直す。

6°. 確定節

A(X, Y)：−B(X, V), B(X, V).

A(X, Y)：−A(X, Y).

の如く，head，bodyに同一のグランドアトムをもつ節を直ちに仮説空間Tから除去する。これは案外，精密化の時間的短縮になる。

7．一つも引数を持たない関数としての定数は精密化操作RFO_3において出来るだけ使用しない方が論理的能力を増加させない仮説が仮説空間に入ってこない。しかし，定数を用いて精密化操作をしないと，矛盾点追跡過程でオラクルに向う回数が増えるので，精密化に使う定数と矛盾点追跡に使う定数とを区別した。

12. 推測Tの局所的最適化

帰納的推論アルゴリズムの段階7での推測Tの局所的最適化は具体的には次のように実施した。

Tからある仮説Aを取り除いたものをT′とする。

(i) T′からS_t内の任意の観測文が導出証明できるならば，TとしてT′を採用する。

(ii) T′からS_t内のある一つの観測文が導出証明できないと判明したならば，TとしてT′を採用せず，Tはもとのままとする。

以上のi，iiをすべてのA∈Tについて実行し，その都度，ゴミの掃除をする。

最適化の実行にあたって考慮した2点を以下に説明する。

1° 仮説空間T内の各確定節については，サイズの大きい確定節ほど前に置くと，矛盾点追跡に時間がかかりにくいなど，都合がよいことがわかった。

2° 帰納的推論アルゴリズムの段階4での2nd whileの繰り返し条件における導出証明の際，真なる観測文を導くのに一度も使用されなかった仮説を他の仮説から区別しておくと，局所的最適化に都合がよい。

13. 合成例

第3章の例3.1の述語

le(X, Y)

をMISに合成させたが，この場合を説明しよう。

1階言語Lの知識をFig.13.1のごとく与えた。1は2つの引数X，Yのデータタイプはint（整数）であり，le(X, Y)はatom（原子論理式）であることを示す。2は，intをデータタイプにもつ入力引数Xの関数S(X)の値のデータタイプはintであることを示す。

```
1 le(int X,int Y),atom
2 int s(int X),func
```

Fig. 13.1　1階言語Lの知識

最初に，6個の事実をFig.13.2のごとく与えると，Fig.13.3のごとく，推測Tが得られた。

正しい推測の一部が第1行目のLrmp[1]に得られており，また，Lrmp[12]において，変数Z，V代りに各々変数X，Yを採用すれば正しい推測の一部となることがわかる。

矛盾点追跡アルゴリズムでは，7個の質問がFig.13.4のごとく，オラクルに対しなされた。そうすると，176個の確定節から成る推測TがFig.13.5のごとく得られた。真理値fmarkがtrueの確定節Lrmp[1]，Lrmp[129]を取り出すと，正しい推測Tが

```
1 le(0,0),true
2 le(s(0),0),false
3 le(0,s(0)),true
4 le(s(0),s(s(0))),true
5 le(s(0),s(s(s(s(s(0))))))),true
6 le(s(s(s(s(s(0))))),s(s(s(s(0))))),false
```

Fig. 13.2　合成開始前に与えた6個の事実

```
サイズkの増加 ... k=3
Lrmp[ 1].fmark=true ,cpt=le(X,X):-
Lrmp[ 2].fmark=fin  ,cpt=le(s(X),Y):-
Lrmp[ 3].fmark=fin  ,cpt=le(X,s(Y)):-
Lrmp[ 4].fmark=fin  ,cpt=le(X,Y):-le(Z,V)
Lrmp[ 5].fmark=false,cpt=le(X,X):-le(Y,Z)
Lrmp[ 6].fmark=false,cpt=le(X,Y):-le(X,Z)
Lrmp[ 7].fmark=false,cpt=le(X,Y):-le(Z,X)
Lrmp[ 8].fmark=false,cpt=le(X,Y):-le(Y,Z)
Lrmp[ 9].fmark=false,cpt=le(X,Y):-le(Z,Y)
Lrmp[10].fmark=false,cpt=le(X,Y):-le(Z,Z)
Lrmp[11].fmark=false,cpt=le(s(X),Y):-le(Z,V)
Lrmp[12].fmark=false,cpt=le(X,s(Y)):-le(Z,V)
Lrmp[13].fmark=false,cpt=le(X,Y):-le(Z,V),le(W,L)
Lrmp[14].fmark=idle ,cpt=le(X,s(X)):-
Lrmp[15].fmark=idle ,cpt=le(s(X),s(Y)):-
Lrmp[16].fmark=idle ,cpt=le(X,s(s(Y))):-
Lrmp[17].fmark=idle ,cpt=le(s(X),X):-
Lrmp[18].fmark=idle ,cpt=le(s(s(X)),Y):-
```

Fig. 13.3　6個の事実から合成された推測T

```
    le(s(0),s(0)) は、真なりや否や？ (t/f)
le(s(0),s(0)) は、真なりや否や？ (t/f)
<t> ok? y/n
le(s(0),s(0)) 新知識　追加
le(s(s(s(s(s(0))))),s(s(s(s(s(0)))))) は、真なりや否や？ (t/f)
<t> ok? y/n
le(s(s(s(s(s(0))))),s(s(s(s(s(0)))))) 新知識　追加
le(s(s(s(0))),s(s(s(s(s(0)))))) は、真なりや否や？ (t/f)
<t> ok? y/n
le(s(s(s(0))),s(s(s(s(s(0)))))) 新知識　追加
le(s(s(s(s(s(0))))),s(s(s(0)))) は、真なりや否や？ (t/f)
<f> ok? y/n
le(s(s(s(s(s(0))))),s(s(s(0)))) 新知識　追加
le(s(s(s(s(s(0))))),0) は、真なりや否や？ (t/f)
<f> ok? y/n
le(s(s(s(s(s(0))))),0) 新知識　追加
le(s(s(s(s(s(0))))),s(0)) は、真なりや否や？ (t/f)
<f> ok? y/n
le(s(s(s(s(s(0))))),s(0)) 新知識　追加
le(s(s(s(s(s(0))))),s(s(0))) は、真なりや否や？ (t/f)
<f> ok? y/n
le(s(s(s(s(s(0))))),s(s(0))) 新知識　追加
```

Fig. 13.4　オラクルに対しなされた７個の質問とその返答

```
Lrmp[  1].fmark=true  ,cpt=le(X,X):-
Lrmp[  2].fmark=idle  ,cpt=le(X,s(X)):-
Lrmp[  3].fmark=idle  ,cpt=le(s(X),s(X)):-
                        :
                        :
Lrmp[128].fmark=false,cpt=le(X,s(X)):-le(X,Y)
Lrmp[129].fmark=true  ,cpt=le(X,s(Y)):-le(X,Y)
Lrmp[130].fmark=false,cpt=le(X,s(s(Y))):-le(X,Z)
                        :
                        :
Lrmp[174].fmark=false,cpt=le(X,s(s(s(s(Y))))):-
Lrmp[175].fmark=false,cpt=le(s(s(s(X))),X):-
Lrmp[176].fmark=false,cpt=le(s(s(s(s(X)))),Y):-
```

Fig. 13.5　６個の事実と７個のオラクルとから合成された推測Ｔ

```
1000 'le(X,X):-.
1001 'le(X,s(Y)):-le(X,Y).
```

Fig. 13.6　得られた正しい推測Ｔとしての合成プログラム

Fig.13.6のごとく得られた。

なお，簡単なリスト処理関数，例えば，

$$\text{member}(X,\ Y)=\begin{cases}\text{true}\cdots X\text{がリスト}Y\text{の要素の場合}\\ \text{false}\cdots\text{その他}\end{cases}$$

も合成させたが，この場合の説明は省略する。

14. むすび

合成されるプログラム（モデルM）がh従順性[1], 3.1節を備えていれば，3条件

(i) 1階言語Lの二つの部分集合である観測言語L_o，仮説言語L_hの対$\langle L_o, L_h\rangle$の許容性[1], 2.2節

(ii) 精密化操作の完全性[1], 5.1節

(iii) 証明操作と精密操作における保守性[1], 5.2節

の下で，MISの帰納的推論アルゴリズムはモデルMを極限において固定する（文献(1), 6.2節を参照)。元来，例からの学習（learning from examples）や帰納的学習（learning by induction）では獲得した知識は原則として完全に検証できないことを思い起こすと，この事実が著者を魅了した理由である。

1階言語Lにおいて

L_o： Lのグランドアトムの集合

L_h： Lのホーン節の集合

とし，証明操作として 導出原理 を採用すると，上述の，iの許容性，iiiの保守性は共に自動的に満足される（文献(1), 2.2節の定理2.3と5.2節とを参照）から，問題は精密化操作における完全性である。

精密化操作RFO_2の適用直後に精密化操作RFO_3を適用する場合と，RFO_3の適用直後にRFO_2を適用する場合に同一の精密化仮説が得られることがあり，この同一性をcheckしながら，第7章の最後で指摘した事実に注目し，完全性を満たす形で，Prologプログラムの合成システムMISをC言語で実現したが，このため，矛盾点追跡を，ホーン節のhead，bodyを入れ換えて導出原理に帰着できた。その結果，合成に必要な所要時間が約1/3程度になった。

問題点は次の通りである。

(イ) 導出原理アルゴリズムにおいては，出現チェック（occur check）は完全には行なっていない。

(ロ) 同様に，導出原理アルゴリズムにおいては，単一化操作でのunifierとして，完全なmguを使っていない。これは事実上，支障なかった。

(ハ) 一度の精密化し過ぎで所要のプログラムが合成不能になることを阻止するために採用された精密化戦略によって生成された，その論理的推論能力がない確定節（仮説）をもゴール節内のアトムとunifiableな確定節の候補としたため，合成時間がかかり過ぎる。

(ニ) 帰納的推論アルゴリズムにおける段階4での全域帰納的関数h(i)を単純な形に選んでいる。

(ホ) これ以上精密化しても仮説空間（理論）の持つ論理的推論能力が増加しないなどの指示を，userが合成過程の途中で与えることが可能な様に会話的にすることが望ましいが，これは現段階では実現していない。

(ヘ) 現在のMISでは仮説空間を主記憶に貯わえているが，これを外部記憶に貯わえておく形式で精密化操作が出来るようにする。

(ト) 極限における固定を有限における同定で事実上置き換えることのできる手段が組み込まれていない。 △

MISの帰納的推論アルゴリズムは精密化（特殊化）の逆の操作である一般化を行なわないために，正の事実例，負の事実例の与えられる順序に対する依存性が大きいMISの帰納的推論能力に長所・短所が現われる。

長所は一般化操作を行なわないため，合成過程の制御が容易になることである。

事実やオラクルに誤りがあったり，言語Lが不十分であると，MISは誤動作したり，所期の効果をもたらさないが，これ以外の短所は，矛盾点を取り除いたとき仮説空間Tの大きさは減少するが，その精密化を仮説空間Tに付加したとき，仮説空間Tの爆発が起こることである。仮説C_1，C_2が共に観測文を導出証明し，C_1がC_2を導出証明する場合，C_2は不要である。このような反例が得られないまま残存している「論理的推論能力をもたない」仮説C_2（ゴミ）が多数存在することになり，仮説空間Tからゴミを取り除く最適化の操作が必ず必要となる。

また，本合成システムには，最適化の操作と実施した後の推測Tを実行効率のよいプログラムに変換する機能は組み込まれていない。

最後に，主張しておきたいことは次の通りである。

(a) $A_i(i=0\sim n)$が同じ述語記号をもつアトムとして，推測T内の確定節

A_0：$-A_1$，A_2，…，A_n.

について，

$size(A_0)\geqq MAX_{i=1\sim n}\ size(A_i)$

であるときのみ，この確定節を精密化の候補とすればよいのではないか。

(b) 推測T中のある確定節がS_t内のどの観測文をも導出証明するのに使用されないならば，この時点ではこの確定節を精密化の候補とする必要はない。

（鈴木昇一・：文教大学情報学部情報システム学科教授，中村三郎：株式会社インターナショナルソフトウェア研究開発課長）

文　献

(1) E.Y. Shapiro：知識の帰納的推論，有川節夫訳，共立出版，1986-07

(2) 鈴木昇一，中村三郎：知識情報処理における帰納的推論，情報研究（文教大学情報学部），Vol.9，1988-12

(3) 有川節夫：知識情報処理の基本技術　3-1　迷語論理と推論，電子通信学会誌，Vol.69，No.11，1986-11，pp.1113-1119

(4) 伊藤貴康，松山隆司：推論ソフトウェアの構成[Ⅱ・完]，電子情報通信学会誌，Vol.70，No.5，1987-05，pp.508-516

(5) Matthew M. Hunt back：An improved version of Shapiro's model inference system, Lect. Notes Commput. Sci.（Univ. Sussex, England），Vol.225，1986，pp.180-187

(6) 阿久津達也，大須賀節雄：Prologに関するいくつかの性質について，人工知能学会誌，Vol.2，No.2，1987-06，pp.223-233

(7) 池田満，溝口理一郎，角所収：HMIS：仮説型モデル推論システム，電子情報通信学会論文誌　D，Vol.J71-D，No.9，1988-09，pp.1761-1771

(8) Hiroki Ishizaka：Model Inference Incorporating Generalization, Journal of Information Processing, Vol.11，No.3，1988，pp.206-211

(9) 沼尾正行，志村正道：説明の部分構造に基づくルール学習法，電子情報通信学会論文誌D-II，Vol. J72-D-II，No.2，1989-02，pp.263-270

(10) 小野浩一：帰納的判断の規定要因の検討——確信反応閾による行動論的分析，心理学研究，Vol.59，No.6，1989，pp.334-341

(11) 佐藤理史，長尾真：文法推論に基づいた翻訳文法の学習方式，コンピュータソフトウェア，Vol.4，No.4，1987-10，pp.352-364