知識工学と人工知能　49－1
（1986. 11. 11）

継承階層 Prolog のメタプログラミングによる最良優先探索システム

赤間　清

（北海道大学　文学部）

継承階層機構を備えた 拡張 Prolog であるＰＡＬを用いて，メタプログラミングにより，最良優先探索システムを与えた．このシステムは，

ａ：確信度付き論理プログラム＋最良優先探索

ｂ：継承階層機構＋継承階層を利用した高速推論

の２つを基礎としており，確信度と継承階層を自由に用いて知識表現を行なうことを可能にする．これは，例えば人間の日常的な行動の世界を模擬するなど，不完全さを含む大量の知識を高速に処理しなければならない意味世界を扱うための基礎システムとして有用である．

Best First Search System by Meta-programming in Inheritance Hierarchy Prolog

Kiyoshi AKAMA

(Faculty of Letters, Hokkaido University, Sappro-shi, 060, Japan)

Logic program with uncertainties (LPU) is a framework which allows us to specify certainties of rules and to compute certainties of inferred conclusions. Best first search algorithm for LPU provides a powerful interpreter that can give, to a given query, the current best one in the remaining solutions (or can tell us the failure of finding one) each time we ask the interpreter for another solution. Important is that each solution is found with the minimal computation required. By utilizing meta-programming technique in inheritance hierarchy (IH) prolog : PAL, we implemented a best first search system. It has two important mechanisms (a) best first search mechanism for LPU, and (b) fast inference mechanism for IH, so that we can represent human everyday knowledge freely by using certainty factors and class names. We demonstrate an availability of the system by an example of narrative story understanding.

1．　はじめに

人間の扱う知識の個々の断片は，それぞれ確実度や信頼度などが異なる．従って個々の知識に確信度を付与することは，人間の多様な知識を扱う上で基本的なものである．それを可能にする有望な枠組みの１つは，確信度付き論理プログラムである．それは，各ホーン節に確信度関数がつけ加えられた論理プログラムである[4)5)]．

確信度関数ｆは，ホーン節を用いて推論をするとき，前提の各論理式の確信度から結論の確信度を計算するために用いられる．それを繰り返すことによって，証明可能なすべての原子論理式に対する確信度が（理論的に）計算できる．従って確信度付き論理プログラムは，任意の原子論理式が証明可能か否かを示すだけでなく，どのような確信度で証明可能かも指定する．

一般に，確信度付き論理プログラムに対する１つの問い合せ（query）がある時，その解は複数個存在し得，それぞれの解は確信度が異なり得る．我々は，より高い確信度を与える証明をより優先して得るようなインタープリタを作るための理論およびその実現について述べた[1)]．そのインタープリタは，先ず最も確信度の高い解を与え，その後は要求に応じて，次に良い解を計算する．それは証明の探索を確信度関数の情報で制御することで達成される．各時点において，インタープリタの計算する量は必要最小限に抑えられている．その探索制御を最良優先探索と呼ぶ．

あるゴールに head がマッチする節が複数個存在する時，そのなかで確信度の高い順に節を選んでゴールを展開することができる．しかしこの方法は，局所的な最適化にはなり得ても，全体の解については，最良のものから求められる保証はない．最良優先探索のアルゴリズムは，大域的最適解を最小のコストで計算する．

最良優先探索あるいはそれに準ずるものを備えることは，知識表現に取って非常に重要である．それは大量の確信度の低い知識に推論を妨げられるのを最小限に抑制するからである．人間の行動などを扱う言語情報処理システム，いろいろな仮説的知識を生成する学習システムなどを含む非常に多くの知識情報処理において，知識はいろいろな確信度の知識の断片の集合体であり，確実度の低い知識の混入は避けられない．最良優先探索システムの存在がなければ，自由な知識表現は在り得ないのである．

本論文では，最良優先探索システムを，継承階層機構を備えた拡張 Prolog であるＰＡＬを用いて，メタプログラミングによって実現する．また，このシステムを利用した知識表現の概略を簡単な例によって示す．それによって，本システムが，最良優先探索機構と継承階層機構の利点を結合し，知識表現にとって非常に有用な環境を提供するための基礎システムとなるものであることを明らかにする．

2．　確信度付き論理プログラムと最良優先探索

確信度付き論理プログラムなどの定義を与える．ここの定義は E. Shapiro の定義[4)]を改訂したものである．ホーン節は，Ａを原子論理式，Ｂをｎ個（$n \geq 0$）の原子論理式の論理積とするとき，Ａ←Ｂの形の節（clause）である．確信度関数ｆは，Ｉを区間 $\{x \mid 0 < x \leq 1\}$ とするとき，I^n からＩへの関数である．確信度付き論理プログラム（logic programs with uncertainties）Ｐは，２項組＜Ａ←Ｂ，ｆ＞の有限集合である．ここで，Ａ←Ｂはホーン節，ｆは確信度関数であり，Ｂに出現する論理式の数とｆの引数の数は等しいものとする．

確信度関数は次のように使われる．ホーン節 $A \leftarrow B_1, B_2, \cdots, B_n$ が代入 θ の適用を受けて，

「もし $B_1\theta, B_2\theta, \cdots, B_n\theta$ ならば $A\theta$ である」

という推論に使われる場合を考える．ホーン節 $A \leftarrow B_1, B_2, \cdots, B_n$ に付随する確信度関数ｆは，$B_1\theta, B_2\theta, \cdots, B_n\theta$ に対する確信度 $b_1, b_2, \cdots, b_n$ から $A\theta$ の確信度ａを

$$a = f(b_1, b_2, \cdots, b_n)$$

によって算出する．代入 θ はホーン節の適用を規定するだけで，確信度の計算には直接関係しない．ここで，特にｎ＝０の場合について注意しておく．そのとき，I^n は，空集合からＩへの写像全体の集合，すなわち，$\{\phi\}$ となる（これはｎ＞０のとき I^n が，$\{1, 2, \cdots, n\}$ からＩへの写像全体の集合と同一視されることの延長として得られる）ので，ｆはただ１つの値 $f(\phi)$ で特

徴づけられ，

$$a = f(\phi)$$

となる．n＝0は節がファクト(fact) Aのばあいであり，任意のθに対して，原子論理式Aθは確信度f（ϕ）で証明可能である．

確信度関数fには次の２条件（置換不変性，単調性）が要請されている．

(1) f（p（X））＝f（X） … pは成分の任意の置換

(2) $X \leqq Y \rightarrow f(X) \leqq f(Y)$

ここで，I^nでの半順序$\leqq$は，I上の全順序$\leqq$により，

$X \leqq Y \leftrightarrow$ 全てのi（$1 \leqq i \leqq n$）に対して，（Xのi成分）$\leqq$（Yのi成分）

と定義されている．

Pにおける原子論理式Aの１つの証明pは，Pの節の有限回の適用からなる．それにそって，各節に付随する確信度関数を順次適用すれば，Aの確信度が求まる．それを，Aのpによる確信度と呼び，c(A，p）で表わす．原子論理式Aが確信度Cで証明可能であるとは，$C \leqq c(A, p)$となるAの証明pが存在することである．

確信度付き論理プログラムのインタープリタを作成するために，最良優先探索のアルゴリズム[1]が考案されている．それは，標準的 Prolog のインタープリタが持つ３つの特徴，

(1) １つの「質問」に対して複数の解を出し得る，

(2) 要請に応じて次の解を出す，

(3) 各時点では，要請に応ずるための最小限の計算しかしない，

を保存し，さらに，

(4) 解は，評価の高い順に出す，

という特徴を持つ．

このアルゴリズムを用いれば，最良の解から順次参照し，参照結果をもとにしてさらに次善の解を参照するか否か決定できる．従って，よりよい説明や仮説の発見などに効果的に用いることができる．また，確信度付き論理プログラムの中に確実度の低い情報が大量に含まれていても，それらをどの程度探索するかをうまく制御できる．

３． 継承階層ＰＲＯＬＯＧ：ＰＡＬによる最良優先探索の実現

最良優先探索の応用範囲を拡大するために，我々は，拡張ＰＲＯＬＯＧ：ＰＡＬによるメタプログラミングによって，最良優先探索システムを実現した（図１），実現方法は畝見の論文[3]を参考にした．このプログラムについて簡単に説明する．ここでは，各ホーン節に付随する確信度関数を１００以下の整数kで表わしており，確信度vの更新の計算を，

$$v' = v + k - 100$$

としている．これは次のような根拠を持つ．もし

$$f(b_1, b_2, \cdots, b_n) = k \times b_1 \times b_2 \times \cdots \times b_n, \quad 0 < k \leqq 1$$

の形の確信度関数を仮定すれば，確信度関数はkの値で特徴づけられる．ここで，両辺の対数をとれば，積の演算は和の演算に置き換えられ，確信度のとり得る区間（０，１］は，新しい区間（－∞，０］に変換される．これを平行移動によって，（－∞，１００］に移し，全てを１００以下の整数の世界で扱うことにすれば，確信度vの更新の計算は，上記のものになる．どちらを取るかは好みの問題だが，１００以下の整数で和差の演算を扱うほうが，確信度を楽に扱えるように思われる．

図１のシステムは，確信度関数付きのホーン節を，

```
(kl k head . body)
```

の形式（kは１００以下の整数）で，知識として受け取る．ここでは，しきい値による枝刈りや，ループ検出などを行なっていない．その意味で，図１は，最も単純な最良優先探索であるが，メタプログラムであるので，これを変更していろいろな機構を付加して，拡張を試みることは比較的容易である．

```
(define bfsc (() (loop (rindc "w>" *q) (if (= *q (end)) (exit)) (c-exec *q))))
(define c-exec
  (((kl . *)) (kl . *))
  (((load *f)) (load *f))
  (((list . ?)) (list db))
  (((retract . ?)) (retract db))
  ((*q)
   (onbt (and (terpri) (print "failure") (terpri)))
   (solve *q *q *a *process)
   (terpri) (print (----------------------> solution))
   (terpri) (print *a)
   (terpri) (pp-form *process)
   (terpri) (rind "more? (y/n) " n) (terpri)))
(define solve
  ((*ansv *goal *ans *process) (demos ((100 (*goal) *ansv () ())) *ans *process)))
(define demos
  (( ((*w () *v ()  *process) . ?) (*w *v) *process))
  (( (*p          . *q) *ans *process)
   (demo *p *next)
   (insert-sort *next *q *newq)
   (cut)
   (demos *newq *ans *process)))
(define demo
  (( (*w () *v () *r) () ) (cut))
  (( (*w () *v (*nextgoal . *stack) *r) *next) (cut)
   (demo (*w *nextgoal *v *stack *r) *next))
  (( (*w (*p) *v *stack *r) *next) (cut)
   (demo1 (*w *p *v *stack *r) *next))
  (( (*w (*p . *q) *v *stack *r) *next) (cut)
   (demo1 (*w *p *v (*q . *stack) *r) *next)))
(define demo1
  (( (*wx (*p . *arg) *v *stack *r) *next)
   (db ? (*p . ?) . ?)
   (cut)
   (accumulate (*ww *body *v *stack ((*wy (*p . *arg) . *body) . *r))
               (and (db *wy (*p . *arg) . *body)
                    (pp-form (*wy (*p . *arg) . *body))
                    (plus *wx *wy -100 *ww))
               *next))
  (( (*w (*p . *arg) *v *stack *r) ((*w () *v *stack ((blt (*p . *arg)) . *r))))
   (exec (*p . *arg))))
(define insert-sort
  ((() * *))
  (((*p . *x) *y *z) (insert-sort:s *p *y *z1) (insert-sort *x *z1 *z)))
(define insert-sort:s
  (((*wx . *px) ((*wy . *py) . *y) ((*wx . *px) (*wy . *py) . *y))
   (greaterp *wx *wy)
   (cut))
  ((*x (*a . *y) (*a . *z)) (insert-sort:s *x *y *z))
  ((*x () (*x))))
(define kl ((*w *h . *b) (asp (db *w *h . *b))))
(define sv ((*w *h . *b) (asp (db *w *h . *b))))
```

図1　拡張ＰＲＯＬＯＧ：ＰＡＬで書いた最良優先探索システム

とくに注意したいのは，ＰＡＬの持つ継承階層機構が，最良優先探索システムの上にそのまま自然に持込まれていることである．すなわち，図１のプログラムでは，継承階層機構については全く考えていないが，後の例に示すように，ＰＡＬの継承階層をそのまま流用して知識が表現できる．これは知識表現を著しく容易にしている．

４．　最良優先探索の応用例

通常の確信度のつかない prolog で扱われている問題で，ホーン節で表わされた各知識に確信度を付与すれば，それは最良優先探索の例になる．従って，応用は無数に考えられる．それらについては別の機会に譲り，ここでは，まだ未完ではあるが，我々がこれから是非考えていきたい種類の問題，すなわち，物語理解を取上げる．

物語理解を次のように単純化して考える．物語に書かれた文の列を

S_1，S_2，…，S_n

とする．読者の知識をＫとする．物語理解とは，読者が列S_1，S_2，…，S_nを，知識Ｋを用いて，うまくつなぐ方法を発見することである．

例えば，図２の漫画の理解に接近するための第１次近似として，次のようなモデル化が可能である．各コマの情報を単純化（あるいは改変）して，次の４つの式で表わす．

```
(kl 100 (action santarou denwa-suru *x))
        ；三太郎は誰かに電話しました．
(kl 100 (action benriya denwa-ukeru *y))
        ；便利屋が誰かの電話を受けました．
(kl 100 (action santarou noru bike1))
        ；三太郎はバイク１に乗りました．
(kl 100 (action benriya ayamaru takeda))
        ；便利屋は武田に謝りました．
```

これが，上の文の列S_1，S_2，…，S_nに当る．改変された主な点は，

(1)：順序（時間）の情報を落とした．
(2)：その他各コマに表現された莫大な情報を落とした．
(3)：３コマ目は簡単のため変更した．

さて，読者の背景知識Ｋは少なくとも図３のような情報を含んでいる必要がある．ここで，背景知識Ｋの記述が，

ａ：確信度付き論理プログラムの採用
ｂ：最良優先探索システムの存在
ｃ：継承階層機構の採用
ｄ：継承階層の表現は高速に推論可能

という４つの事実によって，非常に楽になっていることに注意すべきである．ａは知識間の相対的な確信度の違いを表現可能にしている．またｂは，小さな確信度を持つ知識が大量にあっても，実際の推論の速度を著しく悪

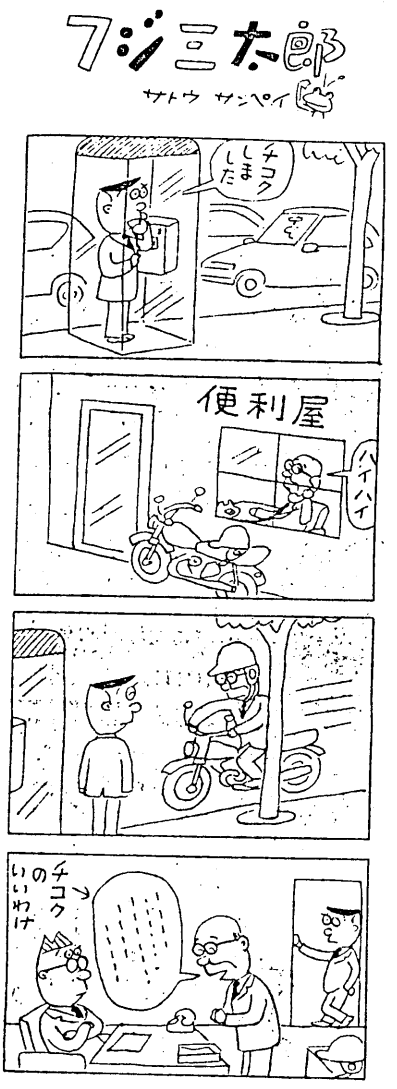

図２　例題の漫画

```
(defc human (kaishain benriya))                    ; 会社員と便利屋は人間です。
(defi kaishain (santarou takeda))                  ; 三太郎と武田は会社員です。
(defi benriya (benriya1))                          ; 便利屋１は便利屋です。
(defc vehicle (bike))                              ; バイクは乗り物です。
(defi bike (bike1))                                ; バイク１はバイクです。
(defc sosiki (kaisha))                             ; 会社は組織です。
(defi kaisha (kaisha1 kaisha2))                    ; 会社１と会社２は会社です。
(defc place (place1 place2))                       ; 場所１と場所２は場所です。

(kl 100 (belong santarou kaisha1))                 ; 三太郎は会社１に属しています。
(kl 100 (belong takeda kaisha1))                   ; 武田は会社１に属しています。
(kl 100 (speed *^bike hayai))                      ; バイクのスピードは速い。
(kl 90
  (action *h^human yatou *b^benriya)               ; 便利屋に連絡して便利屋を雇う。
  (action *h renraku-suru *b))
(kl 80
  (action *h1^human renraku-suru *h2^human)        ; 電話をし、電話を受けて
  (action *h1 denwa-suru *h2)                      ; 連絡がなされる。
  (action *h2 denwa-ukeru *h1))
(kl 70
  (action *h1^human renraku-suru *h2^human)        ; 電話をして連絡する。
  (action *h1 denwa-suru *h2))
(kl 80
  (action *h^human hayaku-iku *p^place)            ; 速い乗り物に乗って速く行く。
  (action *h noru *v^vehicle)
  (speed *v hayai))
(kl 100 (useful hayaku-iku))                       ; 速く行くのは、有益でありうる。
(kl 100 (useful renraku-suru))                     ; 連絡するのは、有益でありうる。
(kl 10  (useful ayamaru))                          ; 謝るのは、有益でありうる。

(kl 70 (state exist *h^kaishain *p^place)          ; 会社員はその会社のある場所にいることが多い。
       (belong *h *k^kaisha)
       (exist *k *p))
(kl 100 (exist kaisha1 place1))                    ; 会社１は場所１にある。

(sv 80
  (action *h^human tikoku-suru *s^sosiki)          ; もし人がある組織に遅刻したら、
  (action *h renraku-suru *o^human)                ; その組織の場所にいる人に連絡する必要がある。
  (exist *s *p^place)
  (state exist *o *p))
(sv 80
  (action *h^human tikoku-suru *s^sosiki)          ; もし人がある組織に遅刻したら、
  (exist *s *p^place)                              ; その組織の所に速く行く必要がある。
  (action *h hayaku-iku *p))
(sv 80
  (action *h^human tikoku-suru *s^sosiki)          ; もし人がある組織に遅刻したら、
  (action *h ayamaru *o^human)                     ; 組織の人に謝る必要がある。
  (belong *o *s))
(sv 20
  (action *h^kaishain *action *object)             ; 会社員は便利屋を雇って、
  (useful *action)                                 ; 自分の代りに何かをやって貰うことができる。
  (action *b^benriya *action *object)
  (action *h yatou *b))
```

図３　背景となる知識の表現例

```
w>(action santarou tikoku-suru kaisha1)                    ;三太郎は会社に遅刻しました。
(------------------------> solution)
(60 (action santarou tikoku-suru kaisha1))                 ;三太郎は会社に遅刻しました。
((100 (speed bike1 hayai))                                 ;彼は会社に速く行く必要があります。
 (100 (action santarou noru bike1))                        ;会社の場所に速く行くために、
 (80 (action santarou hayaku-iku place1)                   ;彼は、スピードの速いバイクに
     (action santarou noru bike1)                          ;乗りました。
     (speed bike1 hayai))
 (100 (exist kaisha1 place1))
 (80 (action santarou tikoku-suru kaisha1) (exist kaisha1 place1)
     (action santarou hayaku-iku place1)))
more? (y/n) y
(------------------------> solution)
(20 (action santarou tikoku-suru kaisha1))                 ;三太郎は会社に遅刻しました。
((100 (exist kaisha1 place1))                              ;彼は、その連絡のために、会社にいる
 (100 (belong santarou kaisha1))                           ;三太郎に電話で連絡しました。
 (70 (state exist santarou place1)                         ;なぜ三太郎に電話したかって？、
     (belong santarou kaisha1)                             ;三太郎は会社の社員だから、きっと
     (exist kaisha1 place1))                               ;会社にいると思ったんです。
 (100 (exist kaisha1 place1))
 (100 (action santarou denwa-suru santarou))
 (70 (action santarou renraku-suru santarou) (action santarou denwa-suru santarou))
 (80 (action santarou tikoku-suru kaisha1)
     (action santarou renraku-suru santarou) (exist kaisha1 place1)
     (state exist santarou place1)))
more? (y/n) y
(------------------------> solution)
(20 (action santarou tikoku-suru kaisha1))                 ;三太郎は会社に遅刻しました。
((100 (exist kaisha1 place1))                              ;彼は、会社の武田に電話で連絡
 (100 (belong takeda kaisha1))                             ;しました。
 (70 (state exist takeda place1) (belong takeda kaisha1) (exist kaisha1 place1))
 (100 (exist kaisha1 place1))
 (100 (action santarou denwa-suru takeda))
 (70 (action santarou renraku-suru takeda) (action santarou denwa-suru takeda))
 (80 (action santarou tikoku-suru kaisha1) (action santarou renraku-suru takeda)
     (exist kaisha1 place1) (state exist takeda place1)))
(------------------------> solution)
(-120 (action santarou tikoku-suru kaisha1))               ;三太郎は会社に遅刻しました。
((100 (belong takeda kaisha1))                             ;彼は会社の武田に謝らねばなりません。
 (100 (action benriya denwa-ukeru santarou))               ;そこで彼は便利屋に電話で連絡し、
 (100 (action santarou denwa-suru benriya))                ;代りに謝ってもらいました。
 (80 (action santarou renraku-suru benriya) (action santarou denwa-suru benriya)
     (action benriya denwa-ukeru santarou))
 (90 (action santarou yatou benriya) (action santarou renraku-suru benriya))
 (100 (action benriya ayamaru takeda))
 (10 (useful ayamaru))
 (20 (action santarou ayamaru takeda) (useful ayamaru)
     (action benriya ayamaru takeda) (action santarou yatou benriya))
 (80 (action santarou tikoku-suru kaisha1) (action santarou ayamaru takeda)
     (belong takeda kaisha1)))
(------------------------> solution)
more? (y/n) n
```

図4　最良優先探索による解

化させることはないことを保証している．特にこの例のように日常の人間の行動に関する知識には，ある場合にだけ成り立つ可能性のあるような知識が多い．ａとｂにより我々は，その様な雑多な知識を，小さな確信度を与えて気楽に書く進めることができる．その際，ｃによって，継承階層が表現でき，しかもｄによって，それが推論の重荷にならないことが保証されていることは，非常に重要である．継承階層機構の存在しない Prolog ではこうはいかない．

なお，図３には，kl だけでなく，sv によって書かれた知識が存在する．図１のシステムでは両者を区別していないが，直観的には，svによって書かれた知識は，kl とは別な種類のものであることは明らかである．これらを区別する扱いについては，別の論文に譲る．

このような準備のもとに，

「三太郎は会社１に遅刻しました．」

から出発して，話の筋をうまく理解する作業は，図４における最良優先探索で（非常に粗く）近似できる．図４では，

(action santarou tikoku-suru kaisha1)

なる問い合せに対する最良優先探索の結果のベスト４が見られる．各解の先頭にはその解の確信度がついている．その下のリストがその解を得るために使われた知識の列である．確信度の計算を最初の解について示すと，

60=100+(100-100)+(100-100)+(80-100)+(100-100)+(80-100)

となる．

５．　むすび

継承階層機構を備えた 拡張 Prolog であるＰＡＬを用いて，メタプログラミングにより，最良優先探索システムを与えた．このシステムは，確信度と継承階層を自由に用いて知識表現を行なうことを可能にする．これは，例えば人間の日常的な行動の世界を模擬するための基礎システムとして便利である．メタプログラミングにより変更，改善は容易であり，今後時間の扱いを含むいろいろな構造を取込む研究に役立つであろう．

言語処理の例を引くまでもなく，意味世界をうまく扱う必要性は非常に高まっている．しかしながら，それに応ずるための研究はまだまだ，不十分である．とくに重要なことは，意味世界を扱うためには，少数の原理を扱うだけではだめで，必然的に大量の知識を高速に処理しなければならないことである．また，必ずしも１００％確実ではない多量の知識の集積を扱わねばならない．そのようなことを可能にする枠組みの開発は，最も重要な研究の１つと考えられる．本論文で与えた枠組みは，ａ：確信度付き論理プログラム＋最良優先探索，ｂ：継承階層機構＋継承階層の高速推論の２つを含んでいる．それらの考え方は，より大規模に意味世界を扱っていく上で，不可欠のものである．

例に示した理解の枠組みは，それ自身議論すべきことがたくさんある．また，仮説生成[2)]などとも関連し，面白く発展する可能性を持っている．それらについては，稿を改めて論じる予定である．

文献

[１] 赤間　清：　最良優先探索ＰＲＯＬＯＧ，情報処理学会知識工学と人工知能研究会，47-8，p57-64（1986）

[２] 赤間　清：　仮説探索システムＨＹＰＯＳＥ，情報処理学会知識工学と人工知能研究会，49-2，p9-18（1986）

[３] 畝見　達夫：　視点としてのデータ型の発見，知識システム方法論夏期シンポジウム報告書，富士通国際研，p.266-282（1986）

[４] Shapiro,E：　"Logic Program with Uncertainties: A tool for Implementing Rule-Based Systems"，　Proc. 8th IJCAI，p529-532（1983）

[５] Sakakibara,Y：　"On Semantics of Logic Programs with Uncertainties"，情報処理学会第３２回全国大会講演論文集，p1245-1246（1986）