知識工学と人工知能　41－12
（１９８５．７．４）

帰納的学習システムの最良応答探索

赤間　清　　（北海道大学　文学部）

１．　はじめに

信念や仮説からなる知識体系では、個々の知識はそれぞれ確実度／信頼度が異なる。そして確実度が高い知識を使うほど、得られる結論の確実度も高くなる。従って、そこでの推論アルゴリズムは確実度のより高い結論を与えるものでなければならない。本論文ではそのような推論アルゴリズム（＝評価順探索アルゴリズム）について述べる。

確実度に差がある知識からなる知識体系を簡単に表現するためには、知識体系の要素である各知識に確実度を表わす重みのようなものを付ければよい。帰納的学習システムＬＳ／０［３，４］の重み付知識表現や、重み付のHORN節集合による知識表現はその例である。以下で考察する評価順探索アルゴリズムが対象とする知識体系は、この種のものである。

帰納的学習システムＬＳ／０の重み付知識表現と、重み付のHORN節集合による知識表現はよく似ている。それは重みを付ける前のそれぞれの知識表現の類似性に由来する。実際、それらを統一する理論的枠組みが存在する［１，２］。本論文では、（その理論の知識は仮定しないが）背後に両者を統一する視点を持って議論を行なう。そのとき、対象となる重み付知識表現を、特に推論アルゴリズムの観点から統一的に見るには、

［ルール］＋［重み］＋［マッチング］

というおおまかな図式が役にたつ。両者のちがいは、［マッチング］のちがいである。

以下の議論は、帰納的学習システムＬＳ／０の重み付知識表現と、重み付のHORN節集合による知識表現の２例を利用して行なう。またHORN節集合による知識表現はよく知られているので、ＬＳ／０の重み付知識表現について扱う場合でも、共通な［ルール］＋［重み］の部分はなるべくHORN節風に記述する。しかし我々が安易にHORN節に依存しているのではない点に注意する必要がある。我々が長期的に求めているのは知識獲得システム、とくに帰納的学習システムの理論である。それから見れば、ここでの知識表現の２つの例は暫定版にすぎない。帰納的学習システムを作成する試みを通して帰納的学習システムに合う知識表現を創り出すことが重要なのである。

２．帰納的学習システムの知識表現と確実度

人間の持つ知識体系の構造のモデル化の試みのひとつに信念システムの研究がある。そこでは、システムの持つ知識は、仮定（assume）型知識と、前提（premise)型の知識に分けられる［５，６］。仮定型の知識だけが変更可能な知識であり、矛盾が生じた場合には、仮定型の知識のいくつかを修正して、それを解消しようとする。

この設定での２種類の知識の差異を、知識の確実度のちがいという観点から見てみると、従来の信念システムの研究では、すべての知識を、絶対に正しくて変更など必要のない知識と、そうとは言えずあとで変更する可能性のある暫定的な知識の２種類に分けていることになる。この仮定は２つの点で疑問が残る。１つは、知識の確実度をた

った２つのレベルに分ける不自然さであり、もう１つは、知識の分類が変化しないことである。知識の確実さはｎとおりあって、しかもそれぞれの確実さは知識獲得が進むにつれて変化していくと考えるのがより自然なモデル化であろう。

知識の確実度に対するより積極的な取り組みは、帰納的学習システムを構成しようとする場合には、緊急の課題となる。ここで帰納的学習システムとは、入力される情報相互間の関連を推定し、それをもとにしてそれらを構造化し自己の知識体系を作り上げ、外界の質問により良く応答するシステムである。帰納的学習システムでは、蓄積される知識の大部分はシステムの推測に基づくものであり、その意味でその知識体系は仮説的知識体系と呼ぶことができる。そのなかの知識には、断片的事実から思いつかれただけの知識もあれば、仮説として作られて以来多くの新しい事実をうまく説明して来た知識もある。それらの知識には異なる確実度を与えて、更新、利用にそれを反映させる必要がある。それらの確実度はつねに新しい事実の挑戦を受けて上下する。また、諸知識の確実度のちがいが新しい知識の生成方法を大きく左右する。帰納的学習システムの場合には、知識の確実度の役割は本質的である。

３．　帰納的学習システムＬＳ／０の応答探索

本節では、ＬＳ／０［３，４］と呼ばれる帰納的学習システムの応答生成のための解探索を題材として、重み付知識表現における推論アルゴリズムに要請される条件を検討する。

帰納的学習システムＬＳ／０が加法の交換法則を学習する場合［７］を考える。ＬＳ／０が、知識がない状態から出発し、簡単な問答によって順に次の４つの情報を得たとする。

「３」は「１＋２＝Ｑ」の答えである。
「９」は「４＋５＝Ｑ」の答えである。
「Ｘ」は「５＋４＝Ｑ」の答えでない。
「９」は「５＋４＝Ｑ」の答えである。

ＬＳ／０はこのような情報を記憶し、それを利用して質問に答える必要がある。従って、ＬＳ／０の扱うべき最小限の知識は、「どのような文字列の質問にはどのような文字列で答えるべきか」を表現したものである。これをＬＳ／０は、知識表現ネットワークの頂点V0に蓄える。しかし頂点V0に得た情報をそのまま書き記すだけでは未知の質問に答える応用力は生じない。ＬＳ／０はネットワークの別の頂点V1，V2，…を生成して、それらの関係をラベルで結びつける。ＬＳ／０の知識表現の中心は、こうしてできる、ラベルで結ばれた頂点の作るネットワークがである。上のような４つの情報が順次与えられたとき、ＬＳ／０は逐次的に知識を更新し、第４の情報を処理した時点で次のような知識を得る。

```
L03 :  100 K2 ( P V0  ( α＋β＝Q, γ))
              ( P V1  ( α, β, γ))
L05 :  100 K1 ( P V1  ( 1, 2, 3))
L06 :    9 K3 ( P V1  ( α, β, γ))
              ( P A1  ( α, β, γ))
              ( N V2  ( α, β, γ))
L09 :  100 K2 ( P V1  ( α, β, 9))
              ( P V3  ( α, β))
L13 :   16 K3 ( P V1  ( α, β, γ))
              ( P V1  ( β, α, γ))
L08 :  100 K1 ( P V2  ( 5, 4, X))
L10 :  100 K1 ( P V3  ( 5, 4))
L11 :  100 K1 ( P V3  ( 4, 5))
L12 :   10 K3 ( P V3  ( α, β))
              ( P A1  ( α, β))
```

ここで、L03 などはラベルであり、V0、V3、A1などは頂点である。またK1などはラベルの型であり、その前の数字は重みである。重みは０から１００までの整数で表わしてあり、大きいほど信頼性が

高いことを意味する。ラベルが何を意味するか簡単に説明するために、ラベルL03 を例にする。これの中心は、「頂点Ｖ１に含まれる文字列の３つ組を（α，β，γ）とするとき、２つ組（α＋β＝Ｑ，γ）は頂点V0に含まれる」というルールである。PROLOG風に書けば次のようになる。

V0(α＋β＝Q，γ) ：－ V1(α，β，γ)．

ただし、α，β，γなどのギリシャ文字は任意の文字列になりうる変数である。ラベルL03 の HEAD の頂点はたまたまV0である。従って、V0が外界の質問と応答の知識を表現するという約束により、ラベルL03 は、「頂点Ｖ１に含まれる文字列の３つ組を（α，β，γ）とするとき、「γ」は「α＋β＝Ｑ」の形の質問の答えである」を意味する。

ここでＬＳ／０に、「２＋１＝Ｑ」という質問が与えられたとする。もしラベルの重みを無視してPROLOGの普通の縦型探索に従えば、L03 , L09 , L12 の各ラベルより、答え「９」を作り出す。これは（ちょうど ASSERTA で各節が定義されたかのように）新しく作られた節ほど先に探索をした場合に（最初に）得られる解である。L12 に現われる頂点A1はシステムの「生得の」知識の１つであり、ラベルで表現される獲得知識とは性格が異なるが、この内容も便宜上節で表現すれば、この解は次の４つの節から作られたものと言える。

```
V0(α＋β＝Q，γ) ：－ V1(α，β，γ)．
V1(α，β，9)     ：－ V3(α，β)．
V3(α，β)        ：－ A1(α，β)．
A1(α，β)．
```

システムはすでに、加法の可換性にあたる仮説（L03 , L13 ）を獲得している。我々が得たい解は、それらと知識L05 を使って作られる答え「３」である。この解には次の３つの節が対応する。

```
V0(α＋β＝Q，γ) ：－ V1(α，β，γ)．
V1(α，β，γ)    ：－ V1(β，α，γ)．
V1(1，2，3)．
```

２つの解の差は、主に、重み１６の仮説L13 を用いるか、または、重み１０の仮説L12 を用いるかによって生じているとみなせる。従って、後者の解を得るには、重みの大きい仮説を用いるほど、得られる解の評価を高くするような解の評価法を定義し、より評価の高い解を求めるアルゴリズムを用いればよい。

解探索アルゴリズムは、単に最も評価の高い解だけを出すのではなく、要請に応じて評価の高い順に順次解を与えることが望ましい。帰納的学習システムの知識の重みは新しい情報に応じて変化させて行くべきものであり、ある時点での重みの小さな差は意味がない場合がある。評価順に複数の解を与えるアルゴリズムがあれば、応答の生成時に評価の高い複数の解を求めて別の基準で比較できる。また、正解が実は次善の解であることがわかったとき、重みだけを直して知識体系を改善するためにも都合がよい。

評価順にすべての解を順次出力するだけなら、あらかじめすべての解を求めておいて、それらを評価順に並べかえれば可能である。しかし我々が、残りの解の中で最も良い解を求めるときの探索時間がより少なくなるようなアルゴリズムを求めているのは言うまでもない。

PROLOG のインタープリタは、プログラムの無限ループを回避できない。PROLOGにおいては、宣言的知識として正しいだけでなく、手続き的に無限ループに陥らないプログラムを書くことがプログラマーの責任である。しかし、このような制約を帰納的学習システムの知識獲得アルゴリズムに課すのは問題がある。知識の獲得は、推論による知識の利用などとは比較にならぬ複雑さをもっている。もしL13 のようなループをつくるのにいちいち探索のための条件チェックをするのでは知識獲得はますます困難になる。帰納的学習システムを

作成するためには、知識の獲得の負担を出来るかぎり軽減することが重要である。これが、解探索アルゴリズムが無限ループを回避する能力を持つべき理由である。しかし無限ループ回避には簡単な完全解決はありえず、よりよい解決のためには、探索に利用できる資源やメタ知識によって探索を制御する問題に踏み込む必要がある。

4. 最良応答探索

4.1 重みつきの知識の探索の問題

PROLOGの各節に重みを付加した知識表現を考える。例えば、次の節集合はその例である。

r1: w1 A←B, C
r2: w2 B←P
r3: w3 B←Q
r4: w4 P←
r5: w5 Q←P
r6: w6 C←

この知識において質問Aに対する解の木は2つある。S式で表わせば、

S1=(r1(r2(r4))(r6))
S2=(r1(r3(r5(r4)))(r6))

となる。

我々の問題は、このような形で与えられた知識を探索して、より良い解を見つけるアルゴリズムを作ることである。そのアルゴリズムは、上の考察に基づけば、

(1) 解の良さを測る評価法が妥当である。
(2) 良い解から順にすべての解を与える。
(3) 最良解を得るまでの探索時間が少ない。
(4) 無限ループを回避できる。
(5) 状況に応じて探索の様子を変更できる。

などの条件を満たすことが望ましい。以下では、この問題に対する1つの解答を示す。

4.2 解の評価

重み付知識体系への質問に対する解の評価について考えるために、解の木に出現する各節の重みを

W1, W2, W3, W4, …, Wn

とする。ただし、木を構成するのにある節が複数回必要な場合には、その数に対応した回数だけ重みも重複して並べるものとする。また簡単のために、重みのとりうる値の範囲を0以上1以下としておく。

解の評価として簡単に思いつくのは、例えば

評価a=W1 *W2 *…*Wn
評価b=MIN(W1, W2, …, Wn)
評価c=MAX(W1, W2, …, Wn)

などである。これらは次の条件(1)を満たす。

(1) 関数Fが存在して、解の評価が、
評価x=F(W1, W2, …, Wn)
と書ける。

さらに、評価aと評価bは次の条件(2)を満たす。

(2) 次の2式を満たす関数Gが存在する。
F(W1, W2, …, Wn+1)
=G(F(W1,W2,…, Wn), Wn+1)
G(x, y) ≦ x

我々はこれらの2つの条件を満たす評価について探索アルゴリズムを与える。このとき関数Gは、評価を順次計算する式として使う。Gを使うたびに評価値は単調に減少する。

4.3 経路探索問題

PROLOG の解探索は、問題の AND結合の書き換えの過程とみなせる。1つの書き換えは1つの節によって行なわれる。例えば、4.1の解T1には、

(A) → (B C) by r1
→ (P C) by r2
→ (C) by r3
→ () by r4

が対応する。

問題の AND結合を頂点、書き換えの節を枝として、すべての解の書き換え過程をまとめた木Tを作る。その木の根は元問題、枝は節、葉は空問題であり、１つの解は根から葉への１つの経路である。木Tにおいては、元問題から同一の節の列の適用によって得られる問題の AND結合は同一頂点に対応させる。しかし逆に、同一の問題 AND結合に対する頂点は同じとは限らない。例えば、空問題は別々の葉となる。解の評価は経路を構成する各枝の重みから、関数Ｆを用いて計算される。この木の上で考えるならば、我々の問題は、根から葉に至る経路を評価の高い順に与える問題である。

木Tの頂点ｎの評価を、Tの根からｎに至る経路を構成する各枝の重みから関数Ｆを用いて算出される数値とする。そのとき、我々の問題はさらに、木Tの葉を評価の高い順に与える問題と言い換えられる。

４．４　評価順探索アルゴリズム

木Tについて、次のような手続きを考える。

１：FRONT ＝｛Tの根｝, PAST ＝｛｝

２：FRONT ＝｛｝ならば、この手続きから出る。

３：FRONT からその元ｎを１つ選ぶ。FRONT からｎを取り除き、かわりにｎの子をすべていれる。またｎを PAST に入れる。すなわち

FRONT ＝FRONT
　　－｛n｝
　　＋［nの子全部の集合］
PAST ＝PAST ＋｛n｝

４：２へ行く。

この手続きを実行中のすべての時点で、次の性質が成り立つ。

「FRONT に含まれる頂点の評価のうちの最大値をＭとする。木TのなかでＭより大きい評価を持つ頂点は PAST または FRONTにのみ存在する。」

これより、求めるアルゴリズムへの要請(2)の前提のもとで、(3)を満たすためには、Ｍを確実に減少させて行けば良いことがわかる。それは、次の方法で可能である。

１：FRONT を出発点ｓと評価値１のペアだけのリストとする。

２：FRONT が空であれば、結果は FAIL であり、この手続きから出る。

３：FRONT から最初の要素（ｎ，ｖ）を除く。

４：ｎが目標頂点ならば、結果は EXIT であり、評価はｖである。ｓからｎに至る経路の情報を得てこの手続きから出る。

５：頂点ｎを展開する任意の枝ｅｉに対して、その重みをWi 、得られる子頂点をｎｉ とし、FRONT にペア

（ｎｉ，Ｇ（ｖ，Wi））

を追加する。

６：FRONT を評価（ペアの第２要素）順に並べかえる。

７：２へ行く。

この手続きは結果として FAIL がでるまで繰り返して呼び出す事ができる。ただし、２回目以降はステップ２から入る。ｍ回目の呼び出しにおいて、その結果が EXIT のときは、Ｆの評価の意味でｍ番目に良い経路を与える。また、結果が FAIL のときは、求める経路はもう存在しない。この探索アルゴリズムを評価順探索アルゴリズムと呼ぶことにする。

４．５　探索の制御

評価順探索アルゴリズムの FRONT を並べかえる方法を変更すれば、いろいろな探索が得られる。例えば、新たに得られた頂点をFRONT のいちばん前にすれば縦型探索が、いちばん後ろにすれば横

型探索が得られる。縦型探索には、最初に解が得られるまでの時間は短く、それに必要な記憶容量も少ないという利点がある。探索対象となる知識体系が大きくなるにつれて、FRONT の並べかえの方法をある程度縦型のやり方に近づけるなどの探索制御が必要になる場合が出て来る。もちろんそのときは、最良の解が求まる保証はない。探索制御には、節の内容や節の重み、問題と節のマッチングの良さなど静的要因を反映させるものと、各時点の利用可能記憶容量や残存探索時間など動的要因を用いるものが考えられる。

ループの回避は、祖先と同一の問題は無条件に失敗させることによって行なう。ここでの祖先は、もちろん、全探索空間の木Tに現われる問題列同士の親子関係を基礎とするものではなく、問題を節によって展開してできる関係である。次の述語を考える。

ex（（a））.
ex（（a | X））　:- ex（X）.

これは無限個の解

（a），（a a），（a a a），…

を持つ。この解を順次すべて取り出すには、親問題ex（X）から、X＝（a | Y）として、子問題ex（Y）を作り出し、この子問題を成功裏に解く必要がある。もしこの２つの問題を同一の問題と判定して、子問題ex（Y）を無条件に失敗させれば、取り出せる解は（a）だけになる。ＬＳ／０では、問題の同一性は変数を含めた完全一致で判定している。この場合、ex（X）とex（Y）は別ものと判定する。

重みが無限ループ回避の助けとなる場合がある。上の例で、ループを作り出す節

ex（（a | X））　:- ex（X）.

の重みが 0.8で、解の評価が

評価a＝W1 ＊W2 ＊…＊Wn

ならば、ループを使うごとに解の評価が低下し、解集合は事実上有限となる。解に対するこのような重み付けは、帰納的学習システムＬＳ／０の知識が有限個の、有限の長さの文字列における推測だという成り立ちにも合う。

帰納的学習システムにおいて、否定の扱いはきわめて重要である。反例を伴わない仮説処理は言語矛盾である。ＬＳ／０の加法の交換法則の学習の例では、第３の情報

「X」は「５＋４＝Q」の答えではない。

が反例であり、この情報の表現のために、ラベル L06 が否定を扱っている。ラベルの中のＰとＮが肯定と否定を区別している。L06 の内容は、

V1（α，β，γ）：-A1（α，β，γ），
not（V2（α，β，γ））.

と書ける。否定が入ると解の評価の枠組みは変更を要する。肯定だけの場合の解は「木の存在」によって指定されるが、否定は「ある探索範囲でのサブ問題の解の非存在」を解に持ち込む。ＬＳ／０の現在の版の評価順探索アルゴリズムでは、あらかじめ与えた解の評価のしきい値によって探索範囲を決定している。これはまた、解の非存在と評価順解出力との重大な矛盾の暫定的解決にもなっている。

５．　文字列マッチング

ＬＳ／０の知識表現ネットワークを探索して、質問から応答を生成するには、文字列マッチングをおこなう必要がある。ここでの文字列マッチングは、排反な３つの集合、定数集合、ｎ文字変数集合、１文字変数集合の要素を有限個並べてできる文字列が２つ与えられたとき、それらを一致させる代入を求めるものである。代入は、ｎ文字変数には任意の定数や変数を含む文字列を対応させるものでよいが、１文字変数に対しては１文字変数か定数を対応させるものでなければならない。

文字列のマッチングはＳ式のマッチングに比してかなり複雑である。マッチングの解が一般には一意には定まらないばかりか、解が無限個存在する場合もある。マッチングのアルゴリズムの複雑さは応答探索の速度を低下させる要因となる。以下では、知識体系を構成するルールに制限を課すことによって必要な文字列マッチングをより簡単な場合だけに留めて、探索を速める方策を示す。

知識体系を構成するルールに対して、

＃「BODY部の各頂点には、引数として、互いに異なる変数しか許さない」

という制限を加える。この制限は例えば、

V0(α＋β＝Q，γ) ：－ V1(α，β，γ)．

など、前掲した全てのルールを許すが、

V2(α) ：－ V3(α，α)．

などは許さない。この種の知識は、ある集合のなかである関係を満たすものだけを取り出して来る働きをする重要なもので、捨てさることはできない。ＬＳ／０の知識表現ではこの問題を、学習アルゴリズムによる知識ネットワークの組み替えによって解決する。例えば上のルールの場合は、次のルールに置き換える。

V2(α) ：－ V4(α)．

ただし頂点V4は新しい頂点で、頂点V3とＫ２型のルール

V3(α，α) ：－ V4(α)．

で結ぶ。Ｋ２型とはルールの意味する包含関係とは逆方向に集合を生成する役割をあわせ持つ型である。学習アルゴリズムは応答生成アルゴリズムの起動に先だって、集合V3のうちから引数の等しいものを抜き出して集合V4を生成しておく必要がある。

評価順探索は

「頂点Ｖに属するＰの形の元を求めよ」

という形の問題が次々に生成され解かれていく過程とみなせる。例えば、ＬＳ／０に対して質問文字列Ｑが与えられたとき解くべき最初の問題では

Ｖ＝ V0

Ｐ＝（Ｑ，α）

となる。ここで問題に対する条件

＊Ｐの引数は定数文字列か変数である

を考える。上の問題はこれを満たす。

ルールのBODYに対する上記の制限＃を仮定するとき、条件＊を満たす問題を解くために生成されるすべてのサブ問題が条件＊を満たすことが言える。この結果必要なマッチングはごく簡単な範囲におさまることがわかる。

６．　評価順探索の帰納的学習への応用の可能性

６．１　重みの変更による学習と評価順探索

質問に対する応答の決定に重みを反映させている場合、重みを変更することによって応答を直接的に改善することができる。まず、応答生成のための評価順探索によって「最良応答」が見つけられ、それが正しいと分かった場合には、その解を強化する。例えば、「最良応答」を与える解の木に出現するルールの重みのうちで最も小さいものを増加させ、２番目に小さい重みと一致させる。「最良応答」が誤りと判明した場合には、重み不適切という原因をまず想定して、評価順探索を続行し次善の応答を次々に求める。そのなかに正しい応答があれば、その応答の評価を高くするようにルールの重みを変更する。次善の応答が存在しない場合には、ルールの生成などを含むもっと大規模な構造変化が必要である。

６．２　構造発見による学習と評価順探索

帰納的学習システムにおいては、複数個の異なる情報を結びつける規則性の発見がつねに行なわれる必要がある。そのなかには評価順探索の枠組

みが有効になる場合がある。例えば、システムが既に２＋１＝３，１＋２＝３，４＋５＝９，１６＋２３＝３９，５＋２＝７などの情報を得ていると仮定する。そのときシスムが更に、

１５＋２４＝３９

という入力を得たとする。もしこの入力を適切に理解できたならシステムは、

mn+pq=rs ← m+p=r,n+q=s

に類似のルールを作る事ができ、それ以後２桁同士の足し算を、１桁同士の足し算に帰着して求めることができる。１５＋２４＝３９という入力を理解し、より一般的なルールを獲得する問題は、１５＋２４から３９を得る過程を、すでに得た他の情報を用いて作り出すことから始まる。我々は、説明可能を意味する述語 expを用いてこれを探索問題に直すことができる。探索は既に獲得したさまざまな重みの推測知識を参照しなければならない。また発見のためのいろいろな heuristics をうまく使い分けなければならない。この探索には少なくとも、ループ回避可能な評価順探索が必要である。

７．　まとめ

確実度や信頼度など、個々の知識の相対比較を表わす重みを表現することは、人間の多様な知識の体系、特に、信念システムや帰納的学習システムを扱う上で基本的なものである。帰納的学習システムＬＳ／０の重み付知識表現や、重み付HORN節集合による知識表現は、そのような知識の表現方法として有望な１つのクラスをなす。そのクラスの知識表現は、特に推論アルゴリズムの観点から見ると、おおまかに

［ルール］＋［重み］＋［マッチング］

の３側面からなる。推論アルゴリズムの探索では、その重みを反映した制御が必要である。解を構成する個々の知識の重みから解の評価を決定するあるクラスの評価関数を仮定して、要請に応じて順次評価順に解を出力するアルゴリズムが与えられた。推論アルゴリズムはマッチングを行ない、その結果できる代入を扱わなければならない。複雑度は対象によって異なり、Ｓ式の場合に比して文字列を対象とするマッチングは探索の負担が大きい。知識の形に制限をつけてマッチングを比較的簡単な範囲に留める方法が与えられた。

これらの考察は帰納的学習システムＬＳ／０を作成する過程で必要となったものであり、その実現に応用されているが、残された課題は多い。本論文で採用した評価順探索の改良に直結するものとしては、知識の内容や利用可能資源に応じた柔軟な探索制御の研究や、完全で高速な文字列マッチングのアルゴリズムの作成などがある。

文献

［１］赤間　清：　連立方程式に基づくネットワークを用いた表現のシステムとその重要なサブクラス，HBSR M 3 1983　北海道大学

［２］赤間　清：　ベキ集合上への自然な拡張によって得られる写像を用いた標号網のクラス　HBSR M 6 1983　北海道大学

［３］赤間　清：　帰納的学習を行なうシステムの初歩的なモデル，HBSR M 7　1985

［４］赤間　清：　帰納的学習システムＬＳ／０を実現するプログラムの概要，HBSR M 8 1985　北海道大学

［５］北上　始，国藤　進，古川康一，宮地泰造：　PROLOGによるBelief System の実現方法，ICOT　Technical Report TM-52，1984

［６］白井英俊：　ＲＭＳ（理由保持機構）を用いた知識表現システム，情報処理学会第２７回全国大会

［７］名塩裕恭：　ヒントを利用する構造化アルゴリズムの実現，北海道大学卒業論文 1984