ICOT Technical Memorandum: TM-0238

TM-0238

類推のメタプログラムに対する部分計算とその応用

赤埴淳一(NTT)
世木博久，古川康一

October, 1986

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

類推のメタプログラムに対する部分計算とその応用

Partial Evaluation of Meta Programs for Analogical Reasoning and its Applications

赤埴　淳一†,世木　博久‡,古川　康一‡

†NTT電気通信研究所,‡ICOT

＜概要＞　複数の対象間の類似性に基いて未知の事実を導き出す類推システムの実現には、類似性に基く事実の推定を効率的に行うことが必要となる。本稿では論理プログラム及び不完全な論理データベースを例にとり、類似性に基いた事実の推定の効率化をはかるために、類推の推論規則に対して部分計算の手法を応用することを提案する。

1.はじめに

我々は通常の演繹では導き出せない未知の事実を得るのに類推をよく用いる。例えばある人の給料がわからないとき、その人の同僚から類推することができる。類推とは一般に、複数の対象間に類似性を発見し、その類似性に基いて何らかの推論を行うことであるが、ここでは類似性に基いて未知の事実を導き出すことを考える。このような類推を計算機に行わせる類推システムの実現には、与えられた対象から類似性を発見する発見機構と、発見された類似性から未知の事実を推定する推定機構が必要となる。類推システムは、発見された類似性に基いて推定機構が推定を行い、さらに推定された事実の妥当性を検証してその事実が妥当でないならば新しい類似性を発見するということを繰返して推論を進めていく。類似性に基く推定は類推の推論規則を与えると演繹的に行うことができるが、類推システムの実現にはこの演繹を効率的に行う必要がある。

本稿では論理プログラム及び不完全な論理データベースを例にとって、類推の推論規則をメタプログラムとして実現し、類似性などに関して部分計算[Takeuchi86]することにより類似性に基く事実の推定の効率化をはかることについて述べる。

2.論理プログラムにおける類推

2つの論理プログラム(確定節の集合)に対して、各プログラムに含まれる定数、関数記号、述語記号の間に対応関係(類似性)が与えられているとする。このとき、対応関係に基いて一方のプログラムで成立する事実から他方のプログラムで成立すると類推される事実を導き出すことを考える。

2.1.論理プログラムに対する類推の推論規則

本節では原口[原口85]に従って事実間の同一視を定義し、それに基いて類推の推論規則[Haraguchi86]を定義する。

P^1,P^2を論理プログラムとする。まず、P^1,P^2の定数間、関数記号間、及び述語記号間の対応関係を次のように定義する。

定義2-1　S-Pairing

$U(P^i)$をP^iのHerbrand領域、$F(P^i),P(P^i)$をそれぞれP^iに出現する関数記号,述語記号の集合とする。このときP^1,P^2の定数間、関数記号間、及び述語記号間の対応関係Ψ_1,Ψ_2,Ψ_3はそれぞれ以下のように定義される。

$$\Psi_1=\{<t,s>\}\subseteq U(P^1)\times U(P^2)$$
$$\Psi_2=\{<f,g>\}\subseteq F(P^1)\times F(P^2)$$
$$\Psi_3=\{<p,q>\}\subseteq P(P^1)\times P(P^2)$$

これらの対応関係の組$\Psi=<\Psi_1,\Psi_2,\Psi_3>$をS-Pairingと呼ぶ。

次に定数間の対応関係を関数記号間の対応関係を用いて拡張する。

定義2-2　Ψ_2によるΨ_1のHerbrand領域への拡張
定数間の対応関係Ψ_1^+を以下のように定義する。

1) $\Psi_1 \subseteq \Psi_1^+ \subseteq U(P^1) \times U(P^2)$
2) $<f,g> \in \Psi_2$,
   $<t_j,s_j> \in \Psi_1^+$,　for $j=1,\cdots,n$
   $\Rightarrow$　$<f(t_1,\cdots,t_n),g(s_1,\cdots,s_n)> \in \Psi_1^+$

S-Pairing $\Psi=<\Psi_1,\Psi_2,\Psi_3>$の下での$P^1$,$P^2$の基礎原子式(以下単にアトムと呼ぶ)間の対応関係は、Ψ_1^+とΨ_3を用いて以下のように定義される。

定義2-3　アトム間の同一視
P^iのHerbrand基底集合を$B(P^i)$とする。P^1,P^2のアトム

$\alpha=p(t_1,\cdots,t_n) \in B(P^1)$,
$\beta=q(s_1,\cdots,s_n) \in B(P^2)$

に対して

$<p,q> \in \Psi_3$,
$<t_j,s_j> \in \Psi_1^+$,　for $j=1;\cdots,n$

が成立するとき、αとβはS-Pairing $\Psi=<\Psi_1,\Psi_2,\Psi_3>$の下で同一視できるといい、

$\alpha\Psi\beta$

と表す。

上記のようにアトム間の同一視を与えると類推の推論規則は以下のように定義できる。

定義2-4　類推の推論規則
P^1の節$A \leftarrow C_1,\cdots,C_n (n>0)$と$P^2$のアトム$\delta_1,\cdots,\delta_n$からS-Pairing Ψに基いてP^2のアトムβを類推する推論規則は以下のような図式で表される。

$$\frac{A \leftarrow C_1,\cdots,C_n}{\alpha \leftarrow \gamma_1,\cdots,\gamma_n}\ (\theta)$$

$$\frac{\alpha \leftarrow \gamma_1,\cdots,\gamma_n \quad (\Psi, M^*(P^1), M^*(P^2))}{\beta \leftarrow \delta_1,\cdots,\delta_n \quad \delta_1,\cdots,\delta_n}$$

$$\beta$$

ここでθは代入であり、最後の推論規則はmodus ponensである。2番目の推論規則はルールの変換と呼ばれ、P^1,P^2の基礎節(ground clause)

$R^1=(\alpha \leftarrow \gamma_1,\cdots,\gamma_n)$,
$R^2=(\beta \leftarrow \delta_1,\cdots,\delta_n)$

に対して

$\alpha\Psi\beta$,
$\gamma_j\Psi\delta_j$,　$\gamma_j \in M^*(P^1)$,　$\delta_j \in M^*(P^2)$,　for $j=1,\cdots,n$

が成立するとき、R^1からR^2を導き出す規則である。また、$M^*(P^i)$はP^iから演繹、或いはS-Pairing Ψに基いて$P^j (j \neq i)$からP^iで類推されるアトムの集合である。

例2-1　次のようなプログラムP^1,P^2とS-Pairing $\Psi=<\Psi_1,\Psi_2,\Psi_3>$が与えられているとする。

$P^1=$ {pressure(pipe1,pres1).
　　flowrate(pipe1,flow1).
　　pipecharacter(Pipe,char(Pres,Flow))←

pressure(Pipe,Pres),flowrate(Pipe,Flow).},

P^2={voltage(resistor1,volt1).
current(resistor1,cur1).
resistance(resistor0,ohm(volt0,cur0)).},

Ψ_1={<pipe1,resistor1>,<pres1,volt1>,
<flow1,cur1>},

Ψ_2={<char,ohm>},

Ψ_3={<pressure,voltage>,<flowrate,current>,
<pipecharacter,resistance>}.

このとき、定義2-2より

<char(pres1,flow1),ohm(volt1,cur1)>$\in\Psi_1^+$

であり、定義2-3より

pressure(pipe1,pres1)Ψvoltage(resistor1,volt1),
flowrate(pipe1,flow1)Ψcurrent(resistor1,cur1),
pipecharacter(pipe1,char(pres1,flow1))Ψ
resistance(resistor1,ohm(volt1,cur1))

であるから、定義2-4からP^2において

resistance(resistor1,ohm(volt1,cur1))

が類推される。

2.2.類推のメタプログラムとその部分計算

前節で定義した類推の推論規則をメタプログラミングした例を図1に示す。定義に現れる各述語の意味は次の通りである。

·get_rule(Alpha,Body,P1):プログラムP1に、Alphaをヘッド、Bodyをボディとするような節が存在することを示す。

·identify(Alpha,Beta,(P1,P2)):P1のアトムAlphaとP2のアトムBetaが同一視できることを示す。すなわちAlphaΨBetaに対応する。

·solve(Gamma,P1):P1でアトムGammaが演繹、或いは類推することができることを示す。すなわちGamma$\in M^*$(P1)であることを示す。

analogize(Beta,P2)はプログラムP2でアトムBetaが類推できることを示す述語である。定義2-4に即していうと、analogizeではプログラムP2のアトムβを導出するために、プログラムP1のルール

$A\leftarrow C_1,\cdots,Cn$

をget_ruleにより選び、適当な代入θにより

$A\theta=\alpha,\ \alpha\Psi\beta$

が成立することをidentifyで調べ、さらに適当なアトム$\delta_1,\cdots,\delta_n$が存在して

$C_j\theta=\gamma_j,\ \gamma_j\Psi\delta_j,$

$\gamma_j\in M^*(P1),\ \delta_j\in M^*(P2),\quad \text{for}\ j=1,\cdots,n$

が成立することをtransform_bodyにより調べる。

このメタプログラムはオブジェクトプログラムに対しインタプリティブに実行されるので実行速度が遅くなる。しかしオブジェクトプログラムとS-Pairingが与えられると、メタプログラムをオブジェクトプログラムとS-Pairingに関して部分計算して特殊化することにより実行効率の向上をはかることができる[Takeuchi86]。例を用いて説明する。図2にオブジェクトプログラムとS-Pairingの例を示す。psi2,psi3はそれぞれ定数,関数記号,述語記号の対応関係を表す述語であり、psi2の第3引数は関数の引数の数を表す。図3に類推のメタプログラムを図2の例に関して部分計算した結果を示す。ここでpsi1_plusは定義2-2で拡張された定数間の対応関係を表す述語である。図3の第3節がp2で類推できるアトムを導出する節であり、図1のメタプログラムに比べて

```
analogize(Beta,P2):-
        get_rule(Alpha,Body,P1),
        P2 \== P1,
        identify(Alpha,Beta,(P1,P2)),
        transform_body(Body,(P1,P2)).
transform_body((Body,Rest_Body),Union):-
        transform_body(Body,Union),
        transform_body(Rest_Body,Union).
transform_body(Gamma,(P1,P2)):-
        solve(Gamma,P1),
        identify(Gamma,Delta,(P1,P2)),
        solve(Delta,P2).
identify(Alpha,Beta,Union):-
        correspond_pred(Alpha,Beta,Union,Num),
        correspond_arg(Alpha,Beta,Union,Num).
correspond_arg(_,_,_,0).
correspond_arg(Alpha,Beta,Union,N):-
        N > 0,
        arg(N,Alpha,Term_A),
        arg(N,Beta,Term_B),
        psi1_plus(Term_A,Term_B,Union),
        N1 is N - 1,
        correspond_arg(Alpha,Beta,Union,N1).
psi1_plus(Term_A,Term_B,(P1,P2)):-
        psi1((P1:Term_A),(P2:Term_B)).
psi1_plus(Term_A,Term_B,Union):-
        correspond_funct(Term_A,Term_B,Union,Num),
        correspond_arg(Term_A,Term_B,Union,Num).
```

図 1　類推のメタプログラム

```
p1:pipecharacter(Pipe,char(Pres,Flow)) :-
        pressure(Pipe,Pres),
        flowrate(Pipe,Flow).
psi2((p1:char),(p2:ohm),Z).
psi3((p1:pressure),(p2:voltage)).
psi3((p1:flowrate),(p2:current)).
psi3((p1:pipecharacter),(p2:resistance)).
```

図 2　オブジェクトプログラムとS-Pairing

```
psi1_plus(A,B,(p1,p2)) :-
     psi1((p1:A),(p2:B)).
psi1_plus(char(A,B),ohm(C,D),(p1,p2)) :-
     psi1_plus(B,D,(p1,p2)),
     psi1_plus(A,C,(p1,p2)).
analogize(resistance(A,B),p2) :-
     psi1_plus(char(C,D),B,(p1,p2)),
     psi1_plus(E,A,(p1,p2)),
     solve(pressure(E,C),p1),
     psi1_plus(C,F,(p1,p2)),
     psi1_plus(E,G,(p1,p2)),
     solve(voltage(G,F),p2),
     solve(flowrate(E,D),p1),
     psi1_plus(D,H,(p1,p2)),
     psi1_plus(E,I,(p1,p2)),
     solve(current(I,H),p2).
```

図 3　特殊化されたメタプログラム

図2のプログラムによく似た構造をしたメタプログラムに特殊化されたものになっている。ここで重要なのは、定数間の対応関係psi1を与えなくとも図3のような特殊化されたメタプログラムが得られることである。すなわち関数及び述語記号の対応関係が与えられれば、部分計算によって特殊化されたメタプログラムが得られるので、定数間のいろいろな対応関係に基く類推をこの特殊化されたメタプログラムを用いて行うことができる。

[N.B.]類推のメタプログラムの問題点とその改良法

図1のメタプログラムには次の2つの問題点がある。

(1) 述語solveの定義の中でanalogizeを呼んでいるので、無限ループに陥る可能性がある。

(2) Ψ_1^+が1対1対応でないとき、類推の意味付けが直感に合わないことが起こりうる。例えば、図3の第3節analogizeの定義を見ると、2番目のリテラルは

$<E,A>\in\Psi_1^+$

5番目のリテラルは

$<E,G>\in\Psi_1^+$

を意味しており、Ψ_1^+が1対1対応でない場合、AとGが異なる定数に束縛される可能性がある。

これらの問題点に対して次のような対応策が考えられる。

(1) 部分計算を行うプログラムPEVAL[Takeuchi86]で用いられているようなループチェックを行う。

(2) (i) SPIC[原口85]のようにΨ_1^+が1対1となるようにΨ_1,Ψ_2を制限する。

(ii) 述語identiifyに対応関係のテーブルをもたせ、Ψ_1^+が1対1となるようにする。

2.3.論理プログラムの類推和との関係

原口[Haraguchi86]は述語記号、関数記号が同じで、定数間に対応関係が与えられているとき、類推を演繹的に行える枠組みを与えている。そこで導入された論理プログラムの類推和とはプログラムに含まれる節を書き換えて得られる節の集合(プログラム)であり、類推は類推和からの演繹として特徴付けられる。本節では、述語記号、関数記号の対応関係がある場合の類推和を定義し、類推のメタプログラムを部分計算して得られた結果と類推和が論理的に同等であることを示す。

S-Pairing $\Psi=<\Psi_1,\Psi_2,\Psi_3>$の下での類推和は以下のように定義される。

定義2-5 PAIR(Ψ_1,Ψ_2)

~を述語記号とする。Ψ_1、Ψ_2に対して次のような節の集合をPAIR(Ψ_1,Ψ_2)で表す。

1) Ψ_1の各要素<t,s>に対して

$t\sim s.$

2) Ψ_2の各要素<f,g>に対して

$f(X_1,\cdots,X_n)\sim g(Y_1,\cdots,Y_n)\leftarrow X_1\sim Y_1,\cdots,X_n\sim Y_n.$

命題1

次の2条件は同値である。

1) $<t,s>\in\Psi_1^+$

2) PAIR(Ψ_1,Ψ_2)$\vdash t\sim s$

すなわち~はΨ_1^+によって定義される定数間の対応関係を表す述語である。

定義2-6 類推和 $P^1\Psi P^2$

論理プログラムP^1,P^2のS-Pairing Ψに関する類推和を$P^1\Psi P^2$で表し、次の5つの集合の和集合で定義する。

1) PAIR(Ψ_1,Ψ_2)

2) P^1の各節のすべての述語に添字1を付けた節の集合

3) P^2の各節のすべての述語に添字2を付けた節の集合

4) P_1の各ルール

$p(X_1,\cdots,X_n)\leftarrow\cdots,q(Y_1,\cdots,Y_k),\cdots.$

を次のように変換した節の集合

$r2(Z_1,\cdots,Z_n) \leftarrow X_1 \sim Z_1,\cdots,X_n \sim Z_n,$
$\cdots$
$Y_1 \sim U_1,\cdots,Y_k \sim U_k,$
$q1(Y_1,\cdots,Y_k),$
$w2(U_1,\cdots,U_k),$
$\cdots$

但し、$<p,r> \in \Psi_3, <q,w> \in \Psi_3$である。

5) P^2についても4)と同様の変換を行った節の集合

例2-2　図2のプログラムの類推和は次のようになる。

$P^1 \Psi P^2$ = {char(A,B)~ohm(C,D)←A~C,B~D.
pipecharacter1(A,char(B,C))←
pressure1(A,B),flowrate1(A,C).
resistance2(A,B)←
E~A,char(C,D)~B,
E~G,C~F,pressure1(E,C),voltage2(G,F),
E~I,D~H,flowrate1(E,D),current2(I,H).}

ここで図3のプログラムと例2-2の類推和を比較してみると非常によく似た構造をしていることがわかる。実際、命題1より図3の述語psi1_plusと例2-2の述語~は論理的に同じものであり、また例2-2の第2節はアトムを演繹できることを示す述語を部分計算すれば得られる。従って結局、図3のプログラムと例2-2の類推和は論理的に同等であることがいえる。すなわち、論理プログラムの類推和は類推のメタプログラムをオブジェクトプログラムに関して部分計算したものと論理的に同等であることが示される。これは原口[Haraguchi86]によって与えられた類推和の別の特徴づけを与えている。

3.不完全なデータベースにおける類推

類推システムの応用として、不完全データベースへの問合わせの問題を扱う。不完全データベース(incomplete database)とは属性値が未知である場合や、不確定である場合のように不完全な情報を持つデータベースのことをいう[Minker 84]。ここでは属性値が未知の場合に類推を応用することを考える。すなわち、あるインスタンスのある属性値が未知のとき、そのインスタンスに類似した他のインスタンスから類推して不完全さを補うことを考える。

3.1.類推の推論規則

不完全なデータベースに対してどのように類推を応用するかを、社員に関する情報を持つデータベースを例にとって説明する。関係「社員」は氏名、年齢、給与、勤務先の4つの属性からなるとする。データベースは「X氏の給与は?」という問い合わせがあると、氏名がXであるインスタンスをさがし、その給与の値を答として返す。しかし、もし、X氏の給与の値が未知であるならば問合せは失敗する。そこで、X氏と類似した他のインスタンスからX氏の給与を類推することを考える。「年齢と勤務先が似ていれば、給与も似ている」ということを知っていれば、X氏の年齢、勤務先に類似した年齢、勤務先を持つA氏の給与から類推できる。すなわち、年齢、給与及び勤務先に関する類似性が与えられ、さらに年齢及び勤務先の類似性と給与の類似性との間に相関関係があるという知識が与えられると、年齢及び勤務先から(他のインスタンスを参考にして)給与を類推することができる。以上述べたことを定式化する。

定義3-1　類似性の相関関係

関係スキーマ$R(X_1,\cdots,X_n)$において、属性X_iの類似性と属性$X_{i_1},\cdots,X_{i_m}$の類似性との間に相関関係があることを次のように表す。

Correlate(R($X_1,\cdots,X_n$),R($Y_1,\cdots,Y_n$),
Sim$_i$(X_i,Y_i),[Sim$_{i_1}$(X_{i_1},Y_{i_1}),$\cdots$,Sim$_{i_m}$(X_{i_m},Y_{i_m})])

ここでSim$_i$(X_i,Y_i)などは属性X_iに関する類似性を表す。

例3-1　関係スキーマ

employee(Name,Age,Salary,Company)

において属性Salaryの類似性と属性Age,Companyの類似性との間に相関関係があるとき、それを

Correlate(employee(Name1,Age1,Sal1,Com1),
employee(Name2,Age2,Sal2,Com2),
sim_sal(Sal1,Sal2),
[sim_age(Age1,Age2),sim_com(Com1,Com2)])

と表す。但し、sim_sal等は類似性を表し、例えば次のように定義される。

sim_sal(X,Y)←diff(X,Y,5000).
sim_age(X,Y)←diff(X,Y,1).
sim_com(X,Y)←same_industry(X,Y).

ここでdiff(X,Y,D)はXとYとの差がD以下であることを示す述語で、same_industry(X,Y)はXとYが同じ業種に属することを示す述語である。

ひとつの属性に関する類似性は一般に複数存在する。上の例でいえば、勤務先に関する類似性は同じ業種である、所在地が同じである等複数の類似性が考えられる。類似性の相関関係の定義に属性に関する類似性を明示するのは複数ある類似性のうちどの類似性を用いるのかを指定する必要があるためである。

定義3-2　インスタンス間の類似性

関係R($X_1,\cdots,X_n$)の2つのインスタンス

α=R($a_1,\cdots,a_n$),
β=R($b_1,\cdots,b_n$)

と類似性の相関関係

Correlate(R($X_1,\cdots,X_n$),R($Y_1,\cdots,Y_n$),
Sim$_i$(X_i,Y_i),[Sim$_{i_1}$(X_{i_1},Y_{i_1}),$\cdots$,Sim$_{i_m}$(X_{i_m},Y_{i_m})])

に対して

sim$_j$(a_j,b_j),　for j=$i_1,\cdots i_m$

が成立するとき、αとβは属性X_iに関してsemi-類似であるといい

$\alpha\Psi_i'\beta$

と表す。さらに

sim$_i$(a_i,b_i)

も成立するとき、αとβは属性X_iに関して類似であるといい

$\alpha\Psi_i\beta$

と表す。

例3-2　例3-1において関係employeeの2つのインスタンス

α=employee(akahani,26,null,NTT),
β=employee(watanabe,25,374000,ICOT)

は属性Salaryに関してsemi-類似である。ここでnullは属性値が未知であることを示す。また、βと

α'=employee(akahani,26,371000,NTT)

は属性Salaryに関して類似である。

このようにインスタンス間の類似性を定義すると類推の推論規則は以下のようになる。

定義3-3　類推の推論規則
関係$R(X_1,\cdots,X_n)$の2つのインスタンス

$\alpha=R(a_1,\cdots,a_n)$,

$\alpha'=R(a_1',\cdots,a_n')$

に対して

$a_i=null$

$a_j=a_j'(j\neq i)$

が成立するとき

$\alpha\approx_i\alpha'$

と表すものとする。i番目の属性値が未知であるαと、αとX_iに関してsemi-類似なβから、i番目の属性値が未知でないα'を導き出す推論規則は次の図式で表される。

$$\frac{\alpha,\ \beta}{\alpha'}\quad(\alpha\approx_i\alpha',\ \alpha'\Psi_i\beta)$$

例3-3　例3-2において

$\alpha\approx_3\alpha'$

であり、αとβから類推の推論規則を用いてα'を導き出すことができる。

3.2.類推のメタプログラムとその応用

前節で定義した類推の推論規則をメタプログラムした例を図4に示す。定義に現れる各述語の意味は以下の通りである。

·get_inst_of_same_rel(Inst1,Inst2):Inst1とInst2は同じ関係のインスタンスであり、かつInst1がデータベースから導出できることを示す。

·unify_except_null(Alpha,Alpha_P,Attr):インスタンスAlphaの属性Attrの値が未知であり、かつAlphaとAlpha_Pが属性Attrを除いて同じ属性値をもつことを示す。すなわち、$Alpha\approx_{Attr}Alpha_P$に対応する。

·similar_fact(Beta,Alpha_P,Attr):インスタンスBetaとAlpha_Pが属性Attrに関して類似であることを示す。すなわち、$Beta\Psi_{Attr}Alpha_P$に対応する。

analogize(Alpha_P)はデータベースからAlpha_Pが類推できることを示す述語である。定義3-3にしたがっていえば、analogizeでは、インスタンスα'を類推するためにget_inst_of_same_relによってα'と同じ関係のインスタンスαとβをデータベースから得、unify_except_nullによって

$\alpha\approx_i\alpha'$

が調べられ、similar_factによって

$\alpha'\Psi_i\beta$

が調べられる。

このメタプログラムを図5の類似性の相関関係に関して部分計算した結果を図6に示す。ここで、fact(Fact)はデータベースからFactが導出できることを示す述語であり、similar(Sim)は類似性Simが存在することを示す述語である。図6のメタプログラムは図4に比べて論理データベースの演繹規則に似た構造をしたメタプログラムに特殊化されており、図4のメタプログラムをインタプリティブに実行するより効率的に実行できる。さらにsim_sal等の類似性の定義について部分計算すれば、より特殊化されたメタプログラムを得ることができる。このように、類推のメタプログラムを属性の類似性、及び類似性の相関関係に関して部分計算を行うことにより、一種のデフォルトの規則を得ることができ、実行効率の向上をはかることができる。

```
analogize(Alpha_P) :-
        get_inst_of_same_rel(Alpha,Alpha_P),
        unify_except_null(Alpha,Alpha_P,Attr),
        get_inst_of_same_rel(Beta,Alpha_P),
        similar_fact(Beta,Alpha_P,Attr).
get_inst_of_same_rel(Inst1,Inst2) :-
        same_relation(Inst2,Inst1),
        fact(Inst1).
unify_except_null(Alpha,Alpha_P,Attr) :-
        contain_null_value(Alpha,Attr),
        unify_except_attr(Alpha,Alpha_P,Attr).
similar_fact(Beta,Alpha_P,Attr) :-
        similar_wrt_attr(Attr,Sim),
        correlate(Beta,Alpha_P,Sim,Sim_List),
        similar_list(Sim_List),
        similar(Sim).
```

図 4 類推のメタプログラム

```
correlate(employee(Name1,Age1,Sal1,Com1),
          employee(Name2,Age2,Sal2,Com2),
          sim_sal(Sal1,Sal2),
          [sim_age(Age1,Age2),sim_com(Com1,Com2)]).
```

図 5 類似性の相関関係

```
analogize(employee(A,B,C,D)) :-
     fact(employee(A,B,null,D)),
     fact(employee(E,F,G,H)),
     similar(sim_age(F,B)),
     similar(sim_com(H,D)),
     similar(sim_sal(G,C)).
```

図 6 特殊化されたメタプログラム

4.まとめ

類推の推論規則をメタプログラムとして実現し、オブジェクトプログラム或いは類似性の定義に関して部分計算することにより、類似性に基く事実の推定を効率的に行えることを、論理プログラム及び不完全なデータベースを例にとって示した。また、論理プログラムに対する類推のメタプログラムを部分計算したものと類推和が論理的に同等なものであることを示した。今後の研究課題としては類似性の発見があげられる。論理プログラムにおける類似性の発見には原口ら[原口86]の研究がある。また横森[Yokomori86]は、データベースへの問合わせ言語間に類似性を発見して問合わせの最適化を行う手法を提案している。本稿で取り上げた不完全なデータベースにおける類推の例についていえば、属性に関する類似性及び類似性の相関関係を発見する手法を確立するのがこれからの課題である。

＜＜謝辞＞＞

本研究の機会を与えて下さったICOT研究所渕一博所長並びにNTT電気通信研究所知識ベース研究室堀内敬之室長、及び熱心に議論していただいたICOT第一研究室の諸氏に感謝致します。

[参考文献]

[Gallaire 84] Gallaire,H., Minker,J. and Nicolas,J.-M., Logic and Databases: A Deductive Approach, *Comput. Surv.*, Vol.16, No.2(1984).

[原口 85] 原口誠,有川節夫,類推の意味論的考察, 知識情報処理セミナー資料, 1985.

[Haraguchi 86] Haraguchi,M. and Arikawa,S.,A Foundation of Reasoning by Analogy: Analogical Union of Logic Programs, *Proc. of Logic Programming Conf.'86*,ICOT,1986.

[原口 86] 原口誠,有川節夫,類推の定式化とその実現,人工知能学会誌,Vol.1,No.1(1986).

[Takeuchi 86] Takeuchi,A. and Furukawa,K., Partial Evaluation of Prolog Programs and its Application to Meta Programming, *Proc. of IFIP-86 Congress*, North-Holland, 1986.

[Yokomori 86] Yokomori,T., On Analogical Query Processing in Logic Database, *Proc. of 12th Int. Conf. on VLDB*, 1986.