ICOT Technical Memorandum: TM-0178

TM-0178

関係型知識ベースにおける単一化結合について

横田治夫(富士通)
森田幸伯，西田健次，伊藤英則

June, 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

知識工学と人工知能 47 － 5
（１９８６．７．１１）

関係型知識ベースにおける単一化結合について

森田　幸伯，横田　治夫，西田　健次，　伊藤　英則

（財）新世代コンピュータ技術開発機構
＊（株）富士通

要約

関係型知識ベース（Relational Knowledge Base)に基づく知識ベースマシンにおけるＲＢＵ演算、特に単一化結合に対する処理方式およびその性質について述べる。関係型知識ベースは、知識ベースに対するひとつのモデルであり、知識は項関係(term-relation)の形で格納・管理される。項関係のなかの項に対しては、単一化に基づく検索、ＲＢＵ:Retrieval-By-Unification)演算により検索を行う。関係型知識ベースは、知識ベースのモデルとして柔軟なモデルであるが、最も単純には単一化結合は、全ての可能なタプルのペアを生成し、そのタプルの指定された属性の値が単一化可能かどうかを調べることになり、その処理は、マシンにとって非常に重い負荷となる。本稿では、単一化結合について効率的な処理を行うための項の順序付けと、そのいくつかの性質について述べる。

Unification-Join Operation On a Relational Knowledge Base

Yukihiro MORITA, Haruo YOKOTA*, Kenji NISHIDA, Hidenori ITOH

Institute for New Generation Computer Technology
*Fujitsu Laboratories Ltd.

This paper describes some properties of retrieval-by-unification (RBU) operations, especially unification-join, on a relational knowledge base, and the ordering of terms. The relational knowledge model is a conceptual model for a knowledge base. Knowledge is represented by term relations in this model. Terms in the relations are retrieved with operations called RBUs (i.e. unification-join and unification-restriction). To perform unification-join in the simplest manner, all possible pairs of tuples in term relations should be checked to see if each pair of terms in the tuple is unifiable or not. This would result in an extremely heavy processing load. The ordering of terms described here improves efficiency considerably.

1．はじめに

第5世代プロジェクトの中期の目的の1つとして知識ベースマシンの開発があげられる。知識情報処理を大規模なシステムで考えた場合、データ処理におけるデータベースのように、知識を効率的に管理・共有できるサブシステムが必要である。本稿では、そのようなサブシステムを効率よく実現するマシンを知識ベースマシンと呼ぶ。

知識ベースマシンは、さまざまなユーザ／ホストが利用するため、その概念構造は基本的で柔軟性に富んでいることが望ましい。〔Yokota 86〕で提案されている関係型知識ベースは、非常に柔軟な知識ベースマシンの概念モデルとなっている。知識を表現する基本要素としては、変数を含むある種の構造体である項(term)を用い、それらの管理の基本要素として関係（relation)を用いる。これらに対して関係代数を拡張したＲＢＵ：Retrieval-By-Unification演算を用意する。項関係で表現された知識は、ホーン節の集合や意味ネットワークモデルによる知識の集合などとみなすことができる。ある形式で格納した項関係は、ホーン節の集合とみなすことができ、その項関係に対して、ＲＢＵ演算を行うことによりinput resolutionを行うことができることを〔Yokota 86〕は示した。〔村上 86〕では、意味ネットワーク的表現ＤＣＫＲ〔小山 85〕の一部を関係型知識ベースモデルの上で実現できることを示した。

しかし、ＲＢＵ演算のうち単一化結合を単純に行うと2つの項関係のすべての可能な組合せに対して指定された属性が単一化可能かどうか調べなければならずマシン側の処理量が非常に大きくなる。

〔森田 86〕では、項の順序付けを用いた組合せの省略による単一化結合の効率化について提案している。

〔大森 86〕では、ハッシュとソートによる単一化結合の効率化を提案している。その方式ではあらかじめ木構造をきめその構造（ハッシュ木）内の定数の情報は可能な限り利用するのでより強く絞り込みがおこなえる。しかし、きめられた構造を用いてハッシングを行うため扱える構造に制限ができる。実際にどちらの方式が優れているかは、絞り込みの後の実際の単一化を処理する方式や全体の実現方式などに依存していると思われる。

本稿では、単一化結合の効率化のための項の順序付けと単一化結合のいくつかの性質について述べる。以下、2章で関係型知識ベースとＲＢＵ演算について数学的な定義を与える。3章では、ＲＢＵ演算に関するいくつかの性質について述べる。

2．関係型知識ベースモデル

データベースでは、ユーザ／アプリケーションの多様な要求に対して、関係モデル(Realtional model)という1つのデータモデルのもつ単純な構造とな操作で柔軟に対応することを可能にした。

知識ベースにおいてもさまざまなユーザ／アプリケーションに対応できるひとつの共通なモデルが必要とおもわれる。関係型知識ベースは、知識ベースマシンに対する1つの概念モデルであり、ユーザのさまざまな知識表現に対応できる単純な構造と操作からなる。

この章では、関係型知識ベースモデルについて、以後の議論に必要な事項について述べる。

2．1．基本概念

関係データモデルでは、領域(domain)D_i上の関係（Relation)を扱う。つまり、関係RはD_i(i=1,n)の直積空間の部分集合と見ることができる。しかし、第一正規型でD_iの要素をアトミック（atomic）なものに制限している。

$$R \subset D_1 \times D_2 \times \cdots \times D_n$$

それに対して、関係型知識ベースモデルでは、領域K_iを項(term)の集合に拡張する。領域Kiの直積空間の部分集合を項関係(term relation)Tと呼ぶ。

$$T \subset K_1 \times K_2 \times \cdots \times K_n$$

つまり、我々は取り扱う対象をデータから変数を含む一種の構造体である項に拡張する。

変数の集合をＶａｒ、関数記号の集合をＦｕｎとし、$Fun \cap Var = \phi$とする。$Fun \cup Var$上の項は、以下のように再帰的に定義される。

1．定数 $a \in F$ は項である。
2．変数 $x \in V$ は項である。
3．もしn引数の関数記号 $f \in F$ があり，$t_1, \ldots, t_n$ が項なら，$f(t_1, \ldots, t_n)$ は項である。
4．上の規則を用いて生成されるものだけが項である。

Term を Fun∪Var 上の項の集合とする。置換 θ :Var→Term は以下のような変数と項の組の順序集合で表現される。

$$\{\{t_i / x_i\} \mid t_i \in Term, x_i \in Var \text{ and if } i \neq j \text{ then } x_i \neq x_j\}$$

置換 θ を項 t に適用した結果を $t\theta$ と書く。$t\theta$ を t のインスタンス(instance)という。項 t_1, t_2 に対し、$t_1\theta = t_2\theta$ を満足する置換 θ が存在する時 t_1, t_2 は単一化可能であるといい、置換 θ を単一化作用素（unifier)という。

項 t_1, t_2 の単一化作用素 θ に対し、項 t_1, t_2 の任意の単一化作用素 θ' に対し、c置換 θ'' が存在し、$\theta' = \theta \cdot \theta''$ であるとき θ は項 t_1, t_2 の最汎単一化作用素(most general unifier)という。

項 t_1, t_2 の最汎単一化作用素を mgu（t_1, t_2）と書く。

項の組の集合 $\{(t_i, u_i) \mid i = 1, \ldots, n\}$ に対し、すべての i に対し、$t_i\theta = u_i\theta$ となる置換 θ をその項の組の同時単一化作用素(simultaneous unifier)という。

同時単一化作用素 θ に対し、任意の同時単一化作用素 θ' に対し、置換 θ'' が存在し、$\theta' = \theta \cdot \theta''$ であるとき θ は最汎同時単一化作用素という。　項の組の集合 $\{(t_i, u_i) \mid i = 1, \ldots, n\}$ の最汎同時単一化作用素を mgsu（$\{(t_1, u_1), \ldots, (t_n, u_n)\}$）と書く。

2．2．RBU

関係データベースのデータ操作言語としては、関係論理(Relational calculus)と関係代数(relational algebra)があった。関係代数が手続き型のシステムであるのに対し、関係論理は非手続き型である。しかし、関係代数のほうがマシンの演算を考えるうえでは考えやすい。そこで、我々は、関係代数を拡張した知識操作言語を考える。

関係データベースから関係型知識ベースへの拡張に伴い、従来の関係代数の結合（join）や制約(restriction）といった演算は、単一化（unification)に基づくものに拡張される。すなわち、結合・制約のなかの等号条件を単一化可能条件に拡張した単一化結合(unification-join)、単一化制約(unification-restriction)を考える。これらを、RBU(Retrcieval-by- unification)演算と呼ぶ。射影（projection)については関係代数のそれと同じである。

タプル μ の属性名の集合Xの各要素の属性の値からなるタプルを $\mu[X]$ と書く。

単一化制約は、項関係のなかから与えられた条件式を満足するタプルの集合をインスタンスの形で求めるものである。条件式は、以下のようなものとする。

$$F = \vee_{k=1,\ldots,n}(\wedge_{j=1,\ldots,m_k}(A_{i_{j,k}} \circ w_{j,k}))$$

だたし、各Λ は属性名をwは属性名または項を示す。wに対してwが属性名ならタプル μ に対するその属性の値を、wが項ならw自身を返す関数take_termを考える。つまり、

$$take\_term(w, \mu) = \begin{cases} \mu(w) & w \text{は属性名} \\ w & w \text{は項.} \end{cases}$$

Λ◇wでタプル μ に対して $\mu[\Lambda]$ とtake_term（w，μ）が単一化可能であるという条件を示す。 Λ,∨はそれぞれ接合（conjunction）選言(disjunction)を意味する。

条件式Fをもつ項関係Tに対する単一化制約の結果関係（$\sigma_F T$ と書く）は以下のようになる。

$$\sigma_F T = \{ \mu\theta \mid \exists \mu \in T, \exists k, \theta = mgsu(\{(\mu[A_{i_{j,k}}], take\_term(w_{j,k}, \mu)) \mid j = 1, \ldots, m_k\}).$$

X，Y，Z，Wをそれぞれ属性名の集合とする。単一化結合は、2つの項関係のタプルの指定された属性の値が単一化可能なタプルのインスタンスの対を合成したタプル集合を求

$ancestor(X,Y) \vee \neg parent(X,Y)$
$ancestor(X,Y) \vee \neg parent(X,Z) \vee \neg ancestor(Z,Y)$
$parent(smith, clark)$
$parent(clark, turner)$
⋮

KB1

$[ancestor(X,Y)\|S]$	$[parent(X,Y)\|S]$
$[ancestor(X,Y)\|S]$	$[parent(X,Z), ancestor(Z,Y)\|S]$
$[parent(smith, clark)\|S]$	S
$[parent(clark, turner)\|S]$	S
⋮	⋮

KB2

$[ancestor(smith,Y)]$	$[parent(smith,Y)]$
$[ancestor(smith,Y)]$	$[parent(smith,Z), ancestor(Z,Y)]$

KB3

$[ancestor(smith, clark)]$	$[parent(smith, clark)]$	$[]$
$[ancestor(smith,Y)]$	$[parent(smith, clark), ancestor(clark,Y)]$	$[ancestor(clark,Y)]$

図１．項関係とＲＢＵ演算の例

めるものである。適当な関数記号を導入することによりμ〔Ｙ〕はＹが属性名の集合の場合でも一つの項とみなせることに注意しよう。

項関係T_1（Ｘ，Ｙ）とT_2（Ｚ，Ｗ）のＹ◇Ｚの条件での単一化結合の結果関係はＸ∪Ｙ∪Ｗ上の関係で以下のようになる。

$$
\begin{aligned}
T_1(X,Y) \underset{Y \diamond Z}{\cdots} T_2(Z,W) \\
= \{ \mu \mid \exists \mu_1 &\in T_1, \exists \mu_2 \in T_2, \\
\theta &= mgu(\mu_1[Y], \mu_2[Z]), \\
\mu[X] &= \mu_1[X]\theta, \\
\mu[Y] &= \mu_1[Y]\theta = \mu_2[Z]\theta, \\
\mu[W] &= \mu_2[W]\theta \quad \}.
\end{aligned}
$$

但し、Ｘ∩Ｗ＝φとし、必要なら属性名は適当に付け替えるものとする。

図１は、項関係の例を示している。

図の上の方に示されたホーン節を表わす項関係がＫＢ１(haed,body)，である。ＫＢ１にhaed◇〔ancestor(smith,W)〕という条件で単一化制約を行った結果がＫＢ２(head,body)である。さらにＫＢ２とＫＢ１をkb2.body◇kb1.headで単一化結合した結果がＫＢ３である。

３．ＲＢＵ演算と項の順序付け

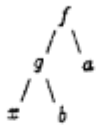
関係モデルは、ユーザに対して柔軟なデータモデルを与えるが、マシンの処理量が大きくなる。特に、大きな関係どうしの結合演算の処理は、非常に大きな計算量を必要とする。そのためＤｅｌｔａ〔Kakuta 85〕では、関係演算専用のハードウェア・関係代数エンジン（ＲＥ）を用いて効率化を計っている。ＲＥはではハードウェアソータを用いて結合(join)などの処理時間の低減をはかっている。

関係型知識ベースモデルにおいても、単一化結合の処理はマシンに対して大きな負担となる可能性がある。単一化結合の効率化が必要である。そのためのＲＢＵ演算と項の順序付けに関するいくつかの性質について述べる。

３．１．項の順序付け

我々は、非常に大量な知識が知識ベースマシンに格納されることを仮定している。その格納場所としては２次記憶を想定している。最も単純に考えれば、単一化結合は２つの項関係（term-relation)のタプル(tuple)のすべての組合せに対して条件を満足するかどうかを調べればよい。しかし、すべての組合せを生成することは、非常に大きな処理負荷がかかる。

そこで関係のタプルを何等かの順序で順序付けることにより組合せを省略したい。



Family order: (f2) (g2) (x) (b0) (a0)
Level order: (f2) (g2) (a0) (x) (b0)

図２．木の線形化表現

[Yokota 86]では項を順序づけるためにジェネラリティ（generality)の概念を定義した。t_1, t_2を項とするとき、t_1がt_2のインスタンスであるときt_2はt_1よりジェネラルであるという。すなわち、

$t_2 \supset t_1$ iff $\exists\theta \quad t_2\theta = t_1$
（但し、θは置換(substitution)）

ジェネラリティによる順序付けは全順序ではない。そこで、ジェネラリティを保存しながら項に対して全順序となる順序付けを考えたい。そのためには、項が木で表現されることに注意し（図２）、その木を線形化して、文字列として表わし、その文字列上の順序を用いれば良い。木の線形化としては、例えば[Knuth 85]によれば家族順(Family order)のものと、レベル順（Level order)のものがある。本稿では、家族順のものを leftmost方式の順序と呼び、レベル順のものをoutermost方式と呼ぶ。m方式による項tの線形化（文字列）表現を$rep_m(t)$と書く。
これらの文字列表現では、各文字はＶａｒ∪Ｆｕｎの要素に対応している。文字cの対応するＶａｒ∪Ｆｕｎの元をnode(c)と書く。
項を順序付けるためにその文字列表現$rep_m(t)$に対して図３に示す辞書的順序を用いる。
ただし、図では以下のような記号を用いている。

文字列ｓのｉ文字目からｊ文字目までをｓ[ｉ；ｊ]で表わす。ただし、ｉ＞ｊのときは空の文字列を表わす。また、length(ｓ)で文字列ｓの長さを表わす。
文字列ｓの中に変数ｘに対応する文字がはじめてでてくる位置をposv(ｓ，ｘ)で表わす。
文字列ｓの中に変数ｘが現れない場合は、

$s_1 > s_2$ if and only if $n = difpos(s_1, s_2)$, and

1. $node(s_1[n;n]), node(s_2[n;n]) \in Var$,
   (a) $posv(s_1, node(s_1[n;n])) > posv(s_2, node(s_2[n;n]))$.
   (b) $posv(s_1, node(s_1[n;n])) = posv(s_2, node(s_2[n;n]))$, $s_1[n+1; length(s_1)] > s_2[n+1; length(s_2)]$.
2. $node(s_1[n;n]) \in Var, node(s_2[n;n]) \in Fun$.
3. $node(s_1[n;n]), node(s_2[n;n]) \in Fun$, $node(s_1[n;n]) > node(s_2[n;n])$ in arbitrary order in Fun

図３．項の文字列表現の辞書的順序

posv(ｓ，ｘ)＝length(ｓ)+1とする。
同様に、文字列ｓの中に変数に対応する文字がはじめてでてくる位置をposv(ｓ)で表わし、現れなければ、posv(ｓ)＝length(ｓ)+1とする。さらに、ｓ[１；posv(ｓ)-1]をprefv(ｓ)と書く。さらに、文字列ｓとｓ'がはじめて違う文字を持つ位置を示すdifpos(ｓ，ｓ')を以下のように定義する。

$$difpos(s_1, s_2) = \begin{cases} 1, & if\ s_1[1;1] \neq s_2[1;1] \text{ or,} \\ & s_1 \text{ or } s_2 \text{ is } null\text{-}string; \\ n, & if\ s_1[1;n-1] = s_2[1;n-1], \\ & s_1[n;n] \neq s_2[n;n]\ (n \geq 2). \end{cases}$$

この２つの方式(left most, outer most)は、ジェネラリティによる順序を保存することが示せる。　２つの方式では、あるノードに対してその子ノードを変化させてもその文字列表現におけるノードに相当する文字より前は変化しない。一般にそのような文字列化の方式ｍの集合をＭとし、以下ｍ∈Ｍであるrep_mについて考える。この場合、置換θは、項の変数のノードをある部分木で置き換えるので
ｋ＝posv(rep_m(ｔ))-1とすると、
$rep_m(t)[1;k] = rep_m(t\theta)[1;k]$
である。
まず、ｍ∈Ｍのとき、rep_mの辞書的順序を用いた項の順序付けは、ジェネラリティを保存することを示す。

定理１
ｍ∈Ｍ、かつｔ⊃ｔ'ならば
$rep_m(t) > rep_m(t')$

［証明］

t コ t' よりある置換 θ が存在して、

$$t\theta = t'$$

任意の置換 θ は'、変数 1 個の置換の合成で表わされることから、変数 1 個に対する置換を $\theta_1 = \{t_1/x_1\}$ とし、

$s = rep_m(t)$, $s' = rep_m(t\theta_1)$

, $k = posv(s, x_1)$

とすると、$s > s'$ を示せばよい。

m∈M より、s と s' は、k − 1 文字目までは一致している。従って、

$$s[1;k-1] = s'[1;k-1]$$

ここで、$s[k;k] \in Var$ であるから

$s'[k;k] \in Fun$ なら $s > s'$、

$s'[k;k] \in Var$ の場合は、t_1 が変数の場合であり $posv(s', t_1) \leqq k$ である。

$posv(s', t_1) < k$ なら

図 3．1(b) より $s > s'$。

$posv(s', t_1) = k$ なら

(1) $posv(s, t_1) = length(s)+1$ の時は $s[k;length(s)]$ と $s'[k;length(s')]$ は、x に相当する文字が t_1 に相当する文字に置き換わっただけで、$posv(x) = posv(t_1) = k$ なので $s = s'$（順序の意味で）である。

(2) $posv(s, t_1) < length(s)+1$ の時は $posv(s, t_1) = k$ とすると、$s[1;k-1] = s'[1;k-1]$ であったから $k \leqq k'$、

$k = k'$ は $x = t_1$ の時だから $s = s'$

$k < k'$ のときは、s, s' の k 文字目がどちらも t_1 で $posv(s, t_1) = k < k' = posv(s', t_1)$ だから $s > s'$

結局、$s > s'$ 故に任意の置換 θ に対しても

$$rep_m(t) \geqq rep_m(t\theta).$$

□

系 1

left most 方式, outer most 方式はジェネラリティを保存する。

定理 2

2 つの項が t, t' に対し m∈M とし、$s = rep_m(t)$, $s' = rep_m(t')$ とするとき t, t' が単一化可能であるなら

$s[1;k-1] = s'[1;k-1]$ (1)

但し、$k = min(posv(s), posv(s'))$

INPUT : Two sets of terms T_1 and T_2.
OUTPUT: All the possible pairs of terms.

Step 1: Set $fs \leftarrow$ *null-string*, $class_i(k) \leftarrow \phi$ for $i = 1$ *or* 2 and $k = 0, 1, 2, \ldots$.

Step 2: Take a term $t \in T_i$, $(i = 1$ *or* $2)$, such that $t \geq t'$ for all $t' \in T_1 \cup T_2$. And let $T_i \leftarrow T_i - \{t\}$. If there are no terms in $T_1 \cup T_2$, then stop.

Step 3: Set $p \leftarrow prefv(rep_m(t))$, and $n \leftarrow difpos(p, fs)$.

Step 4: Let $class_1(k) \leftarrow \phi$, and $class_2(k) \leftarrow \phi$ for $k \geq n$, Let $fs \leftarrow p$.

Step 5: Output all pairs (t, t'), where $t' \in class_j(k)$, $1 \leq k \leq n-1$, $j \neq i$, $(j = 1$ *or* $2)$.

Step 6: Let $class_i(length(p)) \leftarrow class_i(length(p)) \cup \{t\}$ and go to step 2.

図 4．ペア生成アルゴリズム

である。

［証明］

t と t' が単一化可能であるとすると、θ が存在して

$$t\theta = t'\theta$$

$t'' = t\theta = t'\theta$ と置いて、$s'' = rep_m(t'')$ とすると

m∈M、$k \leqq posv(s)$、$k \leqq posv(s')$ だから

$s[1;k-1] = s''[1;k-1] = s'[1;k-1]$

□

3．2．単一化結合における順序付けの効果

項の集合 T_1, T_2 に対し、単一化結合の処理を削減するため単一化の可能性のあるペア (t_1, t_2) $(t_1 \in T_1, t_2 \in T_2)$ を選び出すことを考える。

定理 2 より $t_1 \in T_1$, $t_2 \in T_2$ が単一化可能ならば、$rep_m(t_1)$, $rep_m(t_2)$ について変数に相当する文字が現れるまで両者の前方の文字列は一致している。単一化の可能性のあるペアは、その文字列表現 $rep_m(t_1)$, $rep_m(t_2)$ に対する比較操作だけで選びだすことができる。この選び出す操作は図 4 のアルゴリズムにより容易に行うことができる。

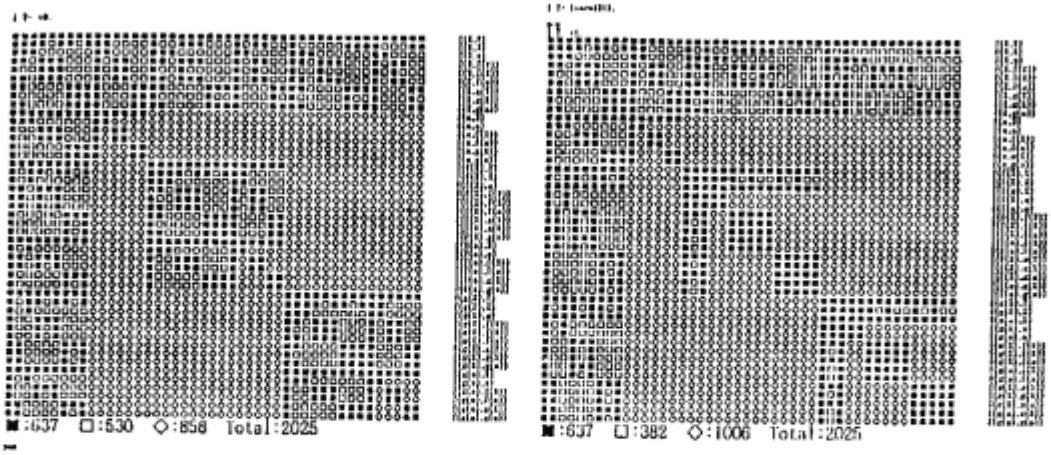


図５．順序付けによる組合せの省略の例

二つの項の集合から順序付けにしたっがて（辞書的順序で大きい順に）マージしながら項を一つずつ取出す。

項tが入力されると「sの値はpref(rep$_{kl}$(t))に変更され、どちらの項集合（T_i）に属していたか及び文字列表現したときの最初に現れる変数の位置kによりclass$_i$(k)に入れられる。

同時に「s[1；k－1)≠ rep$_{kl}$(t)[1；k－1]となるclass$_i$(k)を空集合する。残りのすべての class$_i$の内容とtとは定理２の(1)を満足する。従ってStep5でこれらのうちもう一つの項集合に属していたものとのペアが出力される。

一方、tが入力されたときそれより以前に入力されたt'に対してで定理２の(1)を満足するものがこの時点でどのclassにも入っていないということはない。

もし、どのclass にも含まれていないとすると、t' の属していたclassを空集合にした項t"がt'以降t以前に入力されていたはずである。順序付けに従って順に入力していることからs'＝rep$_{kl}$(t')，s＝rep$_{kl}$(t)
s"＝rep$_{kl}$(t")，k＝posv(t)とおくと

$$s < s'' < s'.$$

t"がt'の属していたclassを空集合にしたことから、あるk'＜kがあって

$$prefv(rep_{kl}(t''))[k';k') \neq prefv(s)[k';k'). \quad (2)$$

ところが、tとt'が定理２の(1)を満足し、s＜s'であるから、
s[1；k－1]＝s'[1；k－1]
k＝posv(t)だからこれらはすべて定数に相当する文字のはずだから(2)に反する。

よって、それより以前に入力された項のうち定理２の(1)を満足するものは、どれかのclassの中にある。

但し、その場合の削減される量は、順序付けにより図５のようになる。

図５は、縦軸、横軸に単一化結合を行う２つの項関係のなかの項を順にならべたものである。最も単純にはこの全平面の相当する項のペアについて単一化可能であるか否かを調べなければならない。図４のアルゴリズムでは定理２の(1)の条件を調べることにより単一化可能かどうかの判定の回数を減らす。図５はトップレベルが２引数関数１種（f），その引数として１引数関数２種（fまたはg）および定数１種（a）からなる項の集合の場合である。

図中■は、単一化可能な項のペアを、◇は条件(1)を満足しないペアを、□は条件(1)は満足するが単一化不可能な項のペアを示す。

項を木で表現した場合、変数は、その葉にしか現れない。従って線形化表現ではレベル優先のものの方が深さ優先のものに比べて変数が後ろに現れやすい。本方式ではposv(rep$_{kl}$(t))だけを調べるのでレベル優先(outer most)の方が絞り込みが強い。

3．3．単一化制約と単一化結合の関係

関係代数である種の制約が結合演算によって行う事ができたように、２．２で述べた条件式を持つ単一化制約は単一化結合で行う事ができる。

例えば、項関係T(Ａ１,Ａ２,Ａ３,Ａ４)に対する単一化制約で、条件式として$\Gamma=(A1 \diamond t1 \cap A2 \diamond A3) \cup (A3 \diamond t2)$を持つ単一化制約の場合、条件を表現する関係として、

T'(Ａ１, Ａ２, Ａ３, Ａ４)＝
{(ｔ１,ｘ２,ｘ２,ｘ４),
(ｙ１,ｙ２,ｔ２,ｙ４)}．

を用意し、お互いの属性(Ａ１,Ａ２,Ａ３,Ａ４)が単一化可能という条件で結合を行えばよい。
つまり

$$\sigma_\Gamma T = T \bowtie T'$$

但しｔ１，ｔ２は項で、ｘ２，ｘ４，ｙ１，ｙ２，ｙ４は変数である。

4．おわりに

関係型知識ベースモデルにおけるＲＢＵ演算および項の順序づけについて、いくつかの性質を述べた。また、項の順序付けにより単一化結合を効率的に行うことができることを示した。

現在は、図４アルゴリズムおよびそれを用いてＲＢＵ演算専用装置（エンジン）[Morita 86]についてシミュレーション等により評価中である。

一方、関係型知識ベースモデルそれ自身の研究やそれを用いた知識ベース管理システムの構築についても現在研究が進められている。

［謝辞］

本検討を進めるにあたり、有益なご示唆を賜った知識ベースマシンＷＧのメンバの方々に感謝致します。

［参考文献］

[Kakuta 85] Kakuta, T., et al., "The Design and Implementation of Relational Database Machine Delta", Proceedings of the International Workshop on Database Machines'85., March 1985.

[knuth 73] knuth, D. E., The Art of Computer Programming, Vol. 1, Fundamental Algorithms, second edition, Addison-Wesley, 1973.

[小山 85] 小山, 田中,"Definite Clause Knowledge Representation", Logic Programming Conference 85, July 1985.

[村上 86] 村上, 他, 「単一化検索言語による知識ベースソフトウェアの記述」, 情報処理学会第３２回全国大会, 1M-9, 1986.

[Murakami 85] Murakami, M., Ykokota, H., Itoh, H., "Formal Semantics of a Relation al Knowledge Base", ICOT TR-149., Dec. 1985.

[森田 86] 森田, 他, 「単一化結合の処理方式」, 情報処理学会第３２回全国大会, 1M-7, 1986.

[Morita 86] Morita, Y., et al., "Retrieval-By-Unification Operation on a Relational Knowledge Base", To appear in Proceedings of the 12th International Conference on VLDB, August 1986.

[大森 86] 大森, 他, 「推論機能と関係データベースの融合－ハッシュとソートによる検索－」, 情報処理学会第３２回全国大会, 1M-6, 1986.

[Yokota 86] Yokota, H., and Itoh, H., "A Model and Architecture for a Relational Knowledge Base", Proceedings of the 13th International Symposium on Computer Architecture, pp. 2-9, June 1986.