(1) 標題

（種別：論文）
項関係上での単一化検索を使ったホーン節推論アルゴリズム*

Resolution Algorithms for Horn Clauses using Retrieval-by-Unification Operation on Term Relations

* 本研究は第五世代コンピュータプロジェクトの一環として行われたものである.

(2) 著者名・所属

横田　治夫*，北上　始，服部　彰
Yokota Haruo, Kitakami Hajime, Hattori Akira

富士通株式会社
FUJITSU LIMITED

会員番号
横田　8230008
北上　8006997
服部　7406896

連絡先
〒211 川崎市中原区上小田中１０１５
富士通研究所人工知能３研
Tel: 044-754-2665 （ダイヤルイン）

(3) 梗概

本稿では，知識ベース処理の１つとして，項関係上での単一化検索 (RBU : Retrieval by Unification)演算の繰り返しによりホーン節推論を行うアルゴリズムを示し，その理論的な定式化を行う．項関係とは，関係データベースにおける関係（テーブル）の格納対象を，述語論理で用いられる項に拡張したものである．また RBU 演算とは，項を取り扱うために関係代数演算の比較処理に単一化を導入し，条件と単一化可能な要素を項関係から検索する演算である．ホーン節を二進木表現にして項関係に格納し， RBU 演算を繰り返すことにより，後ろ向きおよび前向きの推論が可能であることを示す．ホーン節に対する後ろ向き推論の演繹方法としては，Prolog 等で採用されている節中の最左端負リテラルを選択する SLD演繹を対象とし，単一化検索を使って SLD 演繹を実現するアルゴリズムを示す．一方，前向き推論としては，SLD 演繹と同様の選択関数を用いるようにした単位演繹である SUD 演繹を提案し，実現アルゴリズムを提案する．最左端の負リテラルを選択する SLD演繹および SUD 演繹は，健全であり，完全である．対象とするホーン節の集合が充足不能の場合には，提案したアルゴリズムは，それぞれ反駁を求めて停止する．

(4) 本文

1．はじめに

人間の持つ知識を知識ベースとして蓄え，その知識ベースを用いて計算機をより使いやすくすることが重要となってきている．従来，格納する知識の量が増えてきた場合に，大量となった知識をデータとして関係データベースに格納して，推論機構とデータベース機構を結合して全体として知識ベースを用いた推論を行う方式が検討されてきた．その方式は演繹データベースと呼ばれ，盛んに研究が行われている[1),2),3),4)]．しかし，実際に知識を格納しているのが関係データベースであるため，知識を表現するための構造や変数は文字列としてしか格納できないことが問題となる．つまり，推論を行うためには，検索の条件や検索結果を検索の度に変換する必要がある．さらに，構造を反映した効率的な検索も不可能である．

そこで我々は，関係データベースの枠組みを拡張して，格納対象を述語論理における項とするモデルを提案した[5)]．このモデルでは，関係データベースにおける関係（テーブル）に変数を含む再帰的な構造体である項を格納し，項を検索するために関係代数演算に単一化を導入する．格納される知識は推論処理で使われるものと同様の形態であるため，検索時の変換作業が不要になる．また，専用の索引[6)]をつけることにより，構造を反映した効率的な検索が可能となる．さら

に，単純な検索処理の繰り返しのみで推論を実現することができるようになる．

本稿では，その項関係上の単一化検索の繰り返しによるホーン節推論の理論的な定式化を行う．まず，Prolog 等で行われているSLD演繹[7),8)] を項関係上の単一化検索演算の繰り返しで実現するアルゴリズムを示し，Prolog 等異なり完全であることを証明する．次に，SLD演繹が与えられたゴールから後ろ向きに推論を進めるのに対して，現在ある事実から前向きに推論を進める単位演繹[9)] を項関係上の単一化検索の繰り返しにより実現するアルゴリズムを提案する．ただし，一般の単位演繹をそのまま実現するのは困難であるため，SLD演繹と同様の選択関数を用いる単位演繹としてSUD演繹を提案し，SUD演繹を項関係上の単一化検索により実現するアルゴリズムを示す．ホーン節中の最左端の負リテラルを選択するSUD 演繹も， SLD演繹と同様に，健全であり，完全である．

まず，2章で準備として用語の定義を行う．3章では，節を二進木表現することにより，単一化のみで導出形を求めることができることを示す．4章で，項関係上の単一化検索の繰り返しによってSLD演繹を実現するアルゴリズムを示し，ホーン節が充足不能な場合の停止性を証明する．また，全解探索用に変更したアルゴリズムを示す．5章では，SUD演繹を定義して，項関係上の単一化検索によるSUD演繹実現のためのアルゴリズムとその停止性を示す．

2．準備

2．1　項関係

【定義1】F_n をn引数の関数記号(Function Symbol)の有限集合，V を次の式を満足する変数(Variable)の可算無限集合とする．

$$\forall_n, F_n \cap V = \varnothing.$$ ■

【定義2】次のような T を項集合(Term Set)と呼び，その要素 t を項(Term)と呼ぶ．

i)　もし $t \in F_0$ または $t \in V$ ならば $t \in T$

ii)　もし $t_1,...,t_n \in T$, $f \in F_n$ $(n \geq 1)$ ならば

$f(t_1,...,t_n) \in T$. ■

【定義3】 $T_1, T_2,..., T_m$ を項集合としたとき，次のような TR^m をm属性の項関係(Term Relation)と呼ぶ（属性数が自明のときには肩文字は省略する）．

$$T_1 \times T_2 \times ... \times T_m \supset TR^m \ (m \geq 1).$$

このとき次のような tt^m を項タプル(Term Tuple)と呼ぶ．

$$tt^m = (t_1, t_2,..., t_m) \in TR^m.$$

項タプル tt の i 番目のアイテムを $tt[i]$ で表す． ■

2．2　単一化検索

【定義4】 代入(Substitution)λ を $\{t_1 / v_1, ..., t_n / v_n\}$ の形式の有限集合とする．ただし，

$v_1,..., v_n \in V$　（$i \neq j$ のとき $v_i \neq v_j$）

$t_1,..., t_n \in T$　（$t_i \neq v_i$）

とする．代入λによって項t中の変数を置き換えた$t\lambda$をtの代入例(Instance)と呼ぶ ■

【定義5】 $t_1, t_2 \in T$で，$t_1\theta = t_2\theta$なる代入θが存在するとき，t_1, t_2を単一化可能(Unifiable)，代入θをt_1とt_2の単一化作用素(Unifier)と呼ぶ． ■

【定義6】 代入λ_aの後で代入λ_bを行うことを$\lambda_a \bullet \lambda_b$と記す． ■

【定義7】 単一化可能なt_1, t_2に対する単一化作用素の集合をΘとしたとき，

$$\forall\theta_k \in \Theta,\ \exists\lambda_k,\ \theta_k = \sigma \bullet \lambda_k$$

なる単一化作用素$\sigma \in \Theta$を，最汎単一化作用素(Most General Unifier)と呼ぶ． ■

【定義8】 項関係 TR_a^m のi番目$(1 \le i \le m)$と項関係TR_b^nのj番目$(1 \le j \le n)$の属性が単一化可能であるような項タプルの組合せから，次のような新しい項関係TR_c^{m+n}を作る演算を単一化結合 (Unification-Join)と呼び，$uj(TR_a^m, i, TR_b^n, j, TR_c^{m+n})$と記す．

$$tt_c^{m+n} \in TR_c^{m+n}$$

$$\Leftrightarrow \exists\, tt_a^m \in TR_a^m, \exists\, tt_b^n \in TR_b^n, \exists\sigma,\ tt_a^m[i]\sigma = tt_b^n[j]\sigma,$$

$$tt_c^{m+n}[k] = \begin{cases} tt_a^m[k]\sigma & (1 \le k \le m) \\ tt_b^n[k-m]\sigma & (m+1 \le k \le m+n) \end{cases}$$

ここで，σは$tt_a^m[i]$と$tt_b^n[j]$の最汎単一化作用素である． ■

【定義9】 項関係TR_a^mのi番目$(1 \le i \le m)$の属性が変数である項タプルと変数でない項タプルとに分類し

て，次のような新しい項関係 TR_b^m と TR_c^m 作る演算を変数制約(Variable-Restriction)と呼び，$vr(TR_a^m, i, TR_b^m, TR_c^m)$と記す．

$$tt_b^m \in TR_b^m$$
$$\Leftrightarrow {}^{\exists} tt_a^m \in TR_a^m,\ tt_a^m[i] \in V,$$
$$tt_b^m[k] = tt_a^m[k]\ \ (1 \le k \le m)$$
$$tt_c^m \in TR_c^m$$
$$\Leftrightarrow {}^{\exists} tt_a^m \in TR_a^m,\ tt_a^m[i] \notin V,$$
$$tt_c^m[k] = tt_a^m[k]\ \ (1 \le k \le m) \quad \blacksquare$$

【定義１０】 項関係 TR_a^m から，$a_1, \ldots, a_n\ (1 \le a_i \le m)$のn個の属性を取り出して，次のような新しい項関係 TR_b^nを生成する操作を射影(Projection)と呼び，$pr(TR_a^m, [a_1, \ldots, a_n], TR_b^n)$と記す．

$$tt_b^n \in TR_b^n$$
$$\Leftrightarrow {}^{\exists} tt_a^m \in TR_a^m,\ tt_b^n[i] = tt_a^m[a_i]\ \ (1 \le i \le n) \quad \blacksquare$$

【定義１１】項関係 TR_a^m と項関係TR_b^m からその集合和の項関係 TR_c^mを生成する操作を$un(TR_a^m, TR_b^m, TR_c^m)$と記す．この時，変数の付け替えを行う． ■

これらの操作を総称して単一化検索(RBU: Retrieval By Unification)と呼ぶ[5]．この他にも関係代数演算の類推から各種の演算が定義できるが，ここではホーン節推論に使われる演算だけに注目することにする．

3．単一化によるホーン節の導出

ホーン節を二進木構造で表現し，二進木どうしの単一化のみで導出形(Resolvent) が求めれれることを示

す．

【定義１２】二進木述語列を次のように再帰的に定義する．

i)　変数υは二進木述語列である．

ii)　Pが述語，Bが二進木述語列であるとき，

$$\tau(P,B)$$

は二進木述語列である．

ただし，υは述語中に現れない変数，τは述語中に現れない２引数関数記号とする．　■

【定義１３】ホーン節$C=p_0 \vee \neg p_1 \vee \neg p_2 \vee \neg p_3 \vee \ldots \vee \neg p_n$の正リテラルと負リテラルをそれぞれ

$$pos(C)=\tau(p_0,\ \upsilon)$$

$$neg(C)=\tau(p_1,\tau(p_2,\tau(p_3,\ldots\tau(p_n,\upsilon)\ldots)))$$

という別々の二進木述語列にした組を節Cの二進木表現と呼ぶ．υは共通変数，リテラルが空の場合には，変数υをその二進木述語列とする．つまり，Cが正の単位節（１つの正のリテラルのみからなる節）の場合は，

$$neg(C)=\upsilon$$

負節（負リテラルのみからなる節）の場合は，

$$pos(C)=\upsilon$$

空節の場合は，

$$pos(C)=neg(C)=\upsilon$$

となる．　■

【定理１】ホーン節C_1の最左端の負リテラルとホーン節C_2の正リテラルから，C_1とC_2の導出形Rが導出

可能な時，$neg(C_1)$と$pos(C_2)$ の間の単一化により R の二進木表現を得ることができる．

【証明】ホーン節 $C_1 = p_0 \vee \neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n$ の最左端の負リテラル $\neg p_1$ を使って導出形を求めることができるようなもう一方のホーン節 C_2 は，

$$p_1\sigma = q_0\sigma \qquad \text{〔式１〕}$$

であるような，$C_{2a} = q_0 \vee \neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_m\ (1 \leq m)$ もしくは，$C_{2b} = q_0$の何方かである．

（１）C_{2a} の場合

式１から導出形は

$$R_a = (p_0 \vee \neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_m \vee \neg p_2 \vee \ldots \vee \neg p_n)\,\sigma \qquad \text{〔式２〕}$$

となる．C_1の二進木表現は，

$$pos(C_1) = \tau(p_0,\ \upsilon)$$

$$neg(C_1) = \tau(p_1, \tau(p_2, \tau(p_3, \ldots \tau(p_n, \upsilon) \ldots)))$$

C_{2a}の二進木表現は，

$$pos(C_{2a}) = \tau(q_0,\ \upsilon')$$

$$neg(C_{2a}) = \tau(q_1, \tau(q_2, \tau(q_3, \ldots \tau(q_m, \upsilon') \ldots)))$$

ここで，υ，υ' は$p_0 \ldots p_n$,$q_0 \ldots q_m$中に現れない変数である．式１から，$neg(C_1)$と$pos(C_{2a})$は単一化可能であるため，

$$\tau(p_1, \tau(p_2, \tau(p_3, \ldots \tau(p_n, \upsilon) \ldots)))\sigma' = \tau(q_0,\ \upsilon')\sigma'$$

これより，

$$\sigma' = \{\tau(p_2, \tau(p_3, \ldots \tau(p_n, \upsilon) \ldots)) / \upsilon'\} \bullet \sigma \qquad \text{〔式３〕}$$

これを $pos(C_1)$と $neg(C_{2a})$に適用すると，

$$pos(C_1)\sigma'$$

$= \tau(p_0, \upsilon)\sigma'$

$= \tau(p_0, \upsilon)\sigma$　　(∵υ'はp_0に含まれないため)

$neg(C_{2a})\sigma'$

$= \tau(q_1, \tau(q_2, \tau(q_3, \ldots \tau(q_m, \upsilon') \ldots)))\sigma'$

$= \tau(q_1, \tau(q_2, \tau(q_3, \ldots \tau(q_m, \tau(p_2, \tau(p_3, \ldots$

$\tau(p_n, \upsilon) \ldots)))\sigma$

これらは，式２の導出形 R_a の二進木表現に他ならない．

（２）C_{2b}の場合

導出形は，

$R_a = (p_0 \vee \bar{} p_2 \vee \ldots \vee \bar{} p_n)\sigma$　　〔式４〕

となる．また，C_{2b}の二進木表現は，

$pos(C_{2b}) = \tau(q_0, \upsilon')$

$neg(C_{2b}) = \upsilon'$

となる．式１から，$neg(C_1)$と$pos(C_{2b})$は単一化可能であり，$pos(C_{2b})$は$pos(C_{2a})$と全く同じであるから，式３と同一の最汎単一化作用素が求まる．これを$pos(C_1)$と$neg(C_{2b})$に適用すると，

$pos(C_1)\sigma' = \tau(p_0, \upsilon)\sigma'$

$= \tau(p_0, \upsilon)\sigma$

$neg(C_{2a})\sigma' = \upsilon'\sigma'$

$= \tau(p_2, \tau(p_3, \ldots \tau(p_n, \upsilon) \ldots)))\sigma$

これらは，式４の二進木表現と同じである．（証明終了）　■

４．項関係上の単一化検索によるＳＬＤ演繹

SLD演繹とは，Linear resolution with Selection function for Definite clauses の略で，制限された入力演繹の一種である[7]．Prologの処理系は，SLD演繹の一つの実現手段とみなすことができる[8]．SLD演繹では一つの負節 G (つまりPrologのゴール)と確定節（負節以外のホーン節）の集合 D (Prologのプログラム)との和集合$D \cup \{G\}$から反駁を導く．

【定義１４】負節 G_i (ただし，$G_1 = G$とする)の中から選択関数で一つリテラル $\neg p_k$を選び出し，その $\neg p_k$と導出可能であるような適当な節（入力節）C_iを確定節の集合 D の中から選び G_i と C_iの間で導出を行なう．その導出の結果の代入をθ_i，導出形を G_{i+1} とする．G_iが空節になると反駁が存在したことになる．このような，入力節の列 $d_1, d_2, \ldots$，導出形の列 $G_1, G_2, \ldots$，および代入の列 $\theta_1, \theta_2, \ldots$，から得られる反駁をSLD反駁と呼ぶ． ■

【定理２】（SLD反駁の健全性）確定節の集合 D とある負節 G に対して，$D \cup \{G\}$ におけるSLD反駁が存在するならば，$D \cup \{G\}$は充足不能である． ■

【定理３】（SLD反駁の完全性）確定節の集合 D とある負節 G に対して，$D \cup \{G\}$ が充足不能であるとする．この時 $D \cup \{G\}$ におけるSLD反駁が存在する． ■

定理２および定理３の証明については，参考文献[8]を参照されたい．

次に，確定節を二進木表現し一つの確定節 C を一つの項組に対応させて，$pos(C)$ を第一属性，$neg(C)$ を第

二属性として，確定節の集合を項関係に格納し，その上の単一化検索演算の繰り返しによりSLD演繹が行えることを示す．

【アルゴリズム１】

Step 1: 確定節の二進木表現の集合D ($C_k \in D$) の格納されている２属性の項関係をTR_Dとし，

$tt_D \in TR_D$, $tt_D[1] = pos(C_k)$, $tt_D[2] = neg(C_k)$

とする．また，ゴールGも二進木表現にして２属性の項関係TR_Gに

$tt_G \in TR_G$, $tt_G[1] = pos(G) = \upsilon$, $tt_G[2] = neg(G)$

のように格納する．

Step 2: $uj(TR_G, 2, TR_D, 1, TR_{a(0)})$

$pr(TR_{a(0)}, [2,4], TR_{b(0)})$

$i = 0$

Step 3: もし，$TR_{b(i)} = \varnothing$（$TR_{b(i)}$が空集合）なら，

反駁に失敗して終了．

そうでなければ，Step 4 へ．

Step 4: $vr(TR_{b(i)}, 2, TR_{c(i)}, TR_{d(i)})$

Step 5: もし，$TR_{c(i)} \neq \varnothing$なら，反駁に成功して終了，

$TR_{c(i)}$の第一属性がゴールに対する代入を示す．

そうでなければ，Step 6 へ．

Step 6: $i = i + 1$

$uj(TR_{d(i-1)}, 2, TR_D, 1, TR_{a(i)})$

$pr(TR_{a(i)}, [1,4], TR_{b(i)})$

Step 3へ． ■

【定理４】アルゴリズム１は，$D \cup \{G\}$が充足不能で

ある場合には，選択関数で導出形の最左端のリテラルを選ぶSLD反駁を求めて，停止する．

【証明】Step 2の$uj(TR_G, 2, TR_D, 1, TR_{a(0)})$と，定義８から，

$$tt_{a(0)} \in TR_{a(0)}$$
$$\Leftrightarrow \exists tt_G \in TR_G, \exists tt_D \in TR_D, \exists \theta_0,\ tt_G[2]\theta_0 = tt_D[1]\theta_0,$$
$$tt_{a(0)}[1] = tt_G[1]\theta_0,\ tt_{a(0)}[2] = tt_G[2]\theta_0,$$
$$tt_{a(0)}[3] = tt_D[1]\theta_0,\ tt_{a(0)}[4] = tt_D[2]\theta_0$$

これより，

$$neg(G)\theta_0 = tt_G[2]\theta_0 = tt_D[1]\theta_0 = pos(C_0)\theta_0$$

この時，定理１と定義１４より，

$$neg(C_0)\,\theta_0 = neg(G_1)$$

となる．次に，$pr(TR_{a(0)}, [2,4], TR_{b(0)})$から，

$$tt_{b(0)}[1] = tt_{a(0)}[2] = tt_G[2]\theta_0 = neg(G)\,\theta_0$$
$$tt_{b(0)}[2] = tt_{a(0)}[4] = tt_D[2]\theta_0 = neg(C_0)\,\theta_0 = neg(G_1)$$

つまり，$TR_{b(0)}$の第１属性は最初の負節中の変数に対する代入を，$TR_{b(0)}$の第２属性は一番目の導出形の負リテラルを表わしている．次に，Step 4の$vr(TR_{b(i)}, 2, TR_{c(i)}, TR_{c(i)})$と，定義９から，

$$tt_{c(0)} \in TR_{c(0)}$$
$$\Leftrightarrow \exists\, tt_{b(0)} \in TR_{b(0)},$$
$$tt_{c(0)}[2] = tt_{b(0)}[2] = neg(G_1) \in V,$$
$$tt_{c(0)}[1] = tt_{b(0)}[1] = neg(G)\,\theta_0$$
$$tt_{d(0)} \in TR_{d(0)}$$
$$\Leftrightarrow \exists\, tt_{b(0)} \in TR_{b(0)},$$
$$tt_{d(0)}[2] = tt_{b(0)}[2] = neg(G_1) \in V,$$

$$tt_{d(0)}[1] = tt_{b(0)}[1] = neg(G)\,\theta_0$$

もし，$TR_{c(0)}$ が$\varnothing$ なら，Step 6の$uj(TR_{d(i-1)}, 2, TR_D, 1, TR_{a(i)})$，$pr(TR_{a(i)}, [1,4], TR_{b(i)})$ より，同様に，

$$tt_{b(1)}[1] = neg(G)\,\theta_0 \bullet \theta_1$$

$$tt_{b(1)}[2] = neg(C_1)\,\theta_1 = neg(G_2)$$

ただし，

$$neg(G_1)\,\theta_1 = pos(C_1)\,\theta_1$$

さらに，これを繰り返して，

$$tt_{b(i)}[1] = neg(G)\,\theta_0 \bullet \ldots \bullet \theta_i$$

$$tt_{b(i)}[2] = neg(C_i)\,\theta_i = neg(G_{i+1})$$

ただし，

$$neg(G_i)\,\theta_i = pos(C_i)\,\theta_i$$

となる．定義１４と定義１３から，G_i が空節，つまり $pos(G_i) = neg(G_i) = \upsilon$ のときSLD 反駁が得られたことになる．常に $pos(G_i) = \upsilon$ であるため，$neg(G_i) = tt_{b(i-1)}[2] \in V$ つまり，繰り返し中に$TR_{c(i)}$ が$\varnothing$でなくなるとき，導出形として空節が求まったことになる．定理３より $D \cup \{G\}$ が充足不能である場合，空節となる導出形が存在する．Step 2において$uj(TR_{d(i-1)}, 2, TR_D, 1, TR_{a(i)})$で，前段の導出形と導出可能な全ての入力節とから導出形を求めているため，$D \cup \{G\}$ が充足不能である場合には，アルゴリズム１はその反駁に対応する代入の列を求めて終了する．（証明終了） ■

現在実現されているPrologも導出形の最左端のリテラルを選ぶという選択関数で SLD演繹を実現している．しかし，確定節集合 D の中から入力節C_iを探す場

合に，節の書かれた順番に従うため，正確には完全ではない．つまり，$D \cup \{G\}$ が充足不能であっても，反駁が求まらない場合がある．これに対して，アルゴリズム１では，入力節 C_i として単一化結合により可能な全ての候補が適用されるため，$D \cup \{G\}$ が充足不能である場合には必ず SLD反駁が得られる．つまり，アルゴリズム１は SLD演繹を忠実に実現しており，完全性と健全性を保っている．見方を変えると，アルゴリズム１は，横型優先でSLD演繹を行っている．

更に，アルゴリズム１に変更を加えることにより，全解探索を行うアルゴリズム１’を得ることができる．

【アルゴリズム１’】

Step 1: 確定節の二進木表現の集合D ($C_k \in D$) が格納されている２属性の項関係をTR_Dとし，

$tt_D \in TR_D$, $tt_D[1] = pos(C_k)$, $tt_D[2] = neg(C_k)$

とする．

また，ゴール G も二進木表現にして２属性の項関係TR_Gに

$tt_G \in TR_G$, $tt_G[1] = pos(G) = \upsilon$, $tt_G[2] = neg(G)$

のように格納する．

Step 2: $TR_{x(0)} = \varnothing$

$uj(TR_G, 2, TR_D, 1, TR_{a(0)})$

$pr(TR_{a(0)}, [2,4], TR_{b(0)})$

$i = 0$

Step 3: $vr(TR_{b(i)}, 2, TR_{c(i)}, TR_{d(i)})$

$un(TR_{c(i)}, TR_{x(i)}, TR_{x(i+1)})$

Step 4: もし，$TR_{d(i)} = \varnothing$（$TR_{d(i)}$が空集合）なら終了，

$TR_{x(i+1)}$の第一属性が結果を示す．

そうでなければ，Step5 へ．

Step 5: $i = i + 1$

$uj(TR_{d(i-1)}, 2, TR_D, 1, TR_{a(i)})$

$pr(TR_{a(i)}, [1,4], TR_{b(i)})$

Step 3へ． ■

導出形の最左端のリテラルを選ぶ選択関数を持つ SLD 演繹において全ての場合の反駁が有限時間内に得られる場合，アルゴリズム１'はその全ての反駁における変数への代入を求めて停止する．反駁は入力節の選定によりいろいろ有り得るが，アルゴリズム１'では可能な全ての入力節を単一化結合により適用しているため，$TR_{b(i)}$ の第二属性はi番目の繰り返しにおける全ての導出形の集合となる．定義１４から新たな導出形は必ずその前段の導出形から作られる．全ての場合の反駁が有限時間内に得られる場合には，$TR_{d(n)} = \varnothing$ なるnが存在することになる．この時，可能な全ての反駁によって得られる代入は，$TR_{c(i)}\ (1 \le i \le n)$ の第一属性に現れる．$TR_{x(o)} = \varnothing$ と$un(TR_{c(i)}, TR_{x(i)}, TR_{x(i+1)})$より，全代入は$TR_{x(i+1)}$の第一属性に含まれる．

５．項関係上の単一化検索によるＳＵＤ演繹

SLD 演繹が後ろ向きに推論を進めるのに対して前向きに推論を進める方法を考える．前向きの推論方式と

しては単位演繹[9]がある．単位演繹とは，反駁において導出形を求める場合に，導出する一方の節を正の単位節に限定して演繹を進める方式である．しかし，一般の単位演繹を単一化検索を用いて実現することには難がある．そこで，SLD 演繹と同様に，正の単位節と導出を行なう対象のリテラルを選択関数を用いて選ぶ演繹方式を提案し，SUD 演繹（Unit resolution with Selection function for Definite clauses ）と名付ける．本章では，SUD 演繹の定義を行い，さらに単一化検索を用いたSUD 演繹の実現方法を示す．

【定義１５】U_i $(D \supset U_0)$ を正の単位節の集合，H_i $(G \cup (D - U) = H_0)$ を複合節（正の単位節以外のホーン節）の集合とする．複合節の集合H_i中の適当な節C_i の中から選択関数により負リテラル$\neg p_k$を選び，その$\neg p_k$ と正の単位節 $u_k \in U_i$との間で導出を行う．得られた導出形を R_i，代入をθ_i とし，

i) R_iが正の単位節なら，$R_i \in U_{i+1}$

ii) R_i が複合節なら，$R_i \in H_{i+1}$

また，

$U_{i+1} \supset U_i$ かつ $H_{i+1} \supset H_i$

とする．空節のR_i が求まると反駁が存在したことになる．このような，正の単位節の集合の列$U_1, U_2, \ldots$，複合節の集合の列$H_1, H_2, \ldots$，および代入の列$\theta_1, \theta_2, \ldots$，から得られる反駁を<u>SUD 反駁</u>と呼ぶ． ■

【定理５】（SUD反駁の健全性）ホーン節の集合$S = D \cup \{ G \}$ に対してSUD 反駁が存在するならば，S は充足

不能である．　■

【証明】付録で証明する　■

【定理A２】（SUD 反駁の完全性）ホーン節の集合Sが充足不能な時，Sにおいて選択関数として最左端の負リテラルを選ぶようなSUD 反駁が存在する．　■

【証明】付録で証明する　■

次に，U_i 中およびH_i 中の節をそれぞれ３属性の項関係に格納して，単一化検索の繰り返しによりSUD 演繹が行なえることを示す．ここで，３属性の項関係の第一属性は，節中の正リテラルを，第二属性は負リテラルを，第三属性は代入の内容（初期状態の負リテラルに対する代入）を保持するために用いられる．

【アルゴリズム２】

Step 1: 正の単位節の集合($u_k \in U$)の格納されている
３属性の項関係を$TR_{u(0)}$とし，
$tt_{u(0)} \in TR_{u(0)}$, $tt_{u(0)}[1] = pos(u_k)$,
$tt_{u(0)}[2] = tt_{u(0)}[3] = neg(u_k) = \upsilon$
また，複合節の集合($C_l \in H$)の格納されている
３属性の項関係を$TR_{h(0)}$とし，
$tt_{h(0)} \in TR_{hj(0)}$, $tt_{h(0)}[1] = pos(C_l)$,
$tt_{h(0)}[2] = tt_{h(0)}[3] = neg(C_l)$
とする．

Step 2: $i = 0$

Step 3: $uj(TR_{h(i)}, 2, TR_{u(i)}, 1, TR_{s(i)})$
$pr(TR_{u(i)}, [1,5,3], TR_{b(i)})$
$vr(TR_{b(i)}, 2, TR_{tu(i)}, TR_{th(i)})$

Step 4: $vr(TR_{tu(i)}, 1, TR_{c(i)}, TR_{c(i)})$

……もし，$TR_{c(i)} \neq \varnothing$ なら，反駁に成功して終了，

$TR_{c(i)}$の第三属性がゴールに対する代入を示す．

Step 5: $un(TR_{u(i)}, TR_{tu(i)}, TR_{u(i+1)})$

$un(TR_{h(i)}, TR_{th(i)}, TR_{h(i+1)})$

Step 6: もし，$TR_{u(i+1)} = TR_{u(i)}$ かつ $TR_{h(i+1)} = TR_{h(i)}$ なら，

反駁に失敗して終了．

そうでなければ，$i = i + 1$ として Step 3 へ． ■

【定理７】アルゴリズム２は，$D \cup \{G\}$ が充足不能である場合には，導出形の最左端のリテラルを選ぶという選択関数で SUD 反駁を求めて，停止する．

【証明】繰り返し中のStep3の$uj(TR_{h(i)},2,TR_{u(i)},1,TR_{a(i)})$と$pr(TR_{a(i)}, [1,5,3], TR_{b(i)})$ から，

$tt_{b(i)} \in TR_{b(i)}$

$\Leftrightarrow {}^{\exists} tt_{h(i)} \in TR_{h(i)}, {}^{\exists} tt_{u(i)} \in TR_{u(i)}, {}^{\exists}\theta_i,$

$tt_{h(i)}[2]\theta_i = tt_{u(i)}[1]\theta_i,$

$tt_{b(i)}[1] = tt_{h(i)}[1]\theta_i = pos(C_i)\theta_i = pos(R_i)$

$tt_{b(i)}[2] = tt_{u(i)}[2]\theta_i = neg(u_i)\theta_i = neg(R_i)$

$tt_{b(i)}[3] = tt_{h(i)}[3]\theta_i = neg(C_0)\theta_0 \bullet \ldots \bullet \theta_i$

この時，$tt_{h(i)}[2]\theta_i = tt_{u(i)}[1]\theta_i$ から，$neg(C_i)\theta_i = pos(u_i)\theta_i$ となる．また，定義１３より，R_i が単位節なら$neg(R_i)$は変数になる．よって，$vr(TR_{b(i)}, 2, TR_{tu(i)}, TR_{th(i)})$から，$tt_{tu(i)} \in TR_{tu(i)}$ なる$tt_{tu(i)}$は単位節，$tt_{th(i)} \in TR_{th(i)}$ なる$tt_{th(i)}$は複合節を表現する．もし，$vr(TR_{tu(i)}, 1, TR_{c(i)}, TR_{d(i)})$ の $TR_{c(i)}$ が$\varnothing$ ならば，$un(TR_{u(i)}, TR_{tu(i)}, TR_{u(i+1)})$ から $tt_{tu(i)} \in TR_{u(i+1)}$ ，$un(TR_{h(i)}, TR_{th(i)}, TR_{h(i+1)})$ から

$tt_{th(i)} \in TR_{h(i+1)}$として繰り返される．これより，アルゴリズム２は，SUD演繹を実現している．定理５，６より$D \cup \{G\}$が充足不能である場合，またその時だけ，導出形R_jが空節となる．これは，

$$pos(R_j) = neg(R_j) = \upsilon$$

つまり，

$$ttu_j[1] \in V$$

に対応する．すなわち，$D \cup \{G\}$が充足不能である場合，繰り返し中に$TR_{c(i)}$が$\varnothing$でなくなる．そのとき，アルゴリズム２はSUD 反駁に対応する代入の列を求めて終了する．（証明終了） ■

アルゴリズム２の繰り返しでは，新しく導出された単位節や複合節を，前の単位節の集合および複合節の集合に含めて単一化結合を行っている．このため，同一の導出形を何度も作ってしまい，明らかに無駄な処理が多くなる．そこで，新しく作られた導出形とのみ単一化結合を行うように変更したアルゴリズム３を提案する．

【アルゴリズム３】

Step 1: 正の単位節の集合$(u_k \in U)$の格納されている
３属性の項関係を$TR_{u(0)}$とし，
$tt_{u(0)} \in TR_{u(0)}$，
$tt_{u(0)}[1] = pos(u_k)$, $tt_{u(0)}[2] = tt_{u(0)}[3] = neg(u_k) = \upsilon$
また，複合節の集合$(C_i \in H)$の格納されている
３属性の項関係を$TR_{h(0)}$とし，
$tt_{h(0)} \in TR_{h(0)}$，

$tt_{h(0)}[1] = pos(C_i)$, $tt_{h(0)}[2] = tt_{h(0)}[3] = neg(C_i)$

とする.

Step 2: $uj(TR_{h(0)}, 2, TR_{u(0)}, 1, TR_{a(0)})$

$pr(TR_{a(0)}, [1,5,3], TR_{b(0)})$

$vr(TR_{b(0)}, 2, TR_{tu(0)}, TR_{th(0)})$

Step 3: $i = 0$

Step 4: $vr(TR_{tu(i)}, 1, TR_{c(i)}, TR_{d(i)})$

もし，$TR_{c(i)} \neq \varnothing$ なら，反駁に成功して終了，

$TR_{c(i)}$の第三属性がゴールに対する代入を示す.

Step 5: $uj(TR_{h(i)}, 2, TR_{tu(i)}, 1, TR_{ax(i)})$,

$pr(TR_{ax(i)}, [1,5,3], TR_{bx(i)})$,

$vr(TR_{bx(i)}, 2, TR_{ux(i0)}, TR_{hx(i)})$,

$uj(TR_{th(i)}, 2, TR_{u(i)}, 1, TR_{ay(i)})$,

$pr(TR_{ay(i)}, [1,5,3], TR_{by(i)})$,

$vr(TR_{by(i)}, 2, TR_{uy(i0)}, TR_{hy(i)})$,

$uj(TR_{th(i)}, 2, TR_{tu(i)}, 1, TR_{az(i)})$,

$pr(TR_{az(i)}, [1,5,3], TR_{bz(i)})$,

$vr(TR_{bz(i)}, 2, TR_{uz(i0)}, TR_{hz(i)})$,

Step 6: $un(TR_{u(i)}, TR_{tu(i)}, TR_{u(i+1)})$,

$un(TR_{h(i)}, TR_{th(i)}, TR_{h(i+1)})$,

$un(TR_{ux(i)}, TR_{uy(i)}, TR_{uw(i)})$,

$un(TR_{uw(i)}, TR_{uz(i)}, TR_{tu(i+1)})$

$un(TR_{hx(i)}, TR_{hy(i)}, TR_{hw(i)})$,

$un(TR_{hw(i)}, TR_{hz(i)}, TR_{h(i+1)})$

Step 7: もし，$TR_{u(i+1)}=TR_{u(i)}$ かつ $TR_{h(i+1)}=TR_{h(i)}$ なら，

反駁に失敗して終了.

そうでなければ，$i = i + 1$としてStep 4へ． ■

【定理7】アルゴリズム3は，アルゴリズム2と同様に$D \cup \{G\}$が充足不能である場合には，導出形の最左端のリテラルを選ぶという選択関数でSUD反駁を求めて，停止する．

【証明】ここでは，$uj(TR_a^m, i, TR_b^n, j, TR_c^{m+n})$を$TR_c^{m+n} = TR_a^m \ {}_i\&_j \ TR_b^n$，$un(TR_a^m, TR_b^m, TR_c^m)$を$TR_c^m = TR_a^m \cup TR_b^m$で表わすことにする．単一化結合は，集合の要素間の組合せであるから，

$$TR_{h(i+1)} \ {}_2\&_1 \ TR_{u(i+1)}$$
$$= (TR_{h(i)} \cup TR_{th(i)}) \ {}_2\&_1 \ (TR_{u(i)} \cup TR_{tu(i)})$$
$$= (TR_{h(i)} \ {}_2\&_1 \ TR_{u(i)}) \cup (TR_{h(i)} \ {}_2\&_1 \ TR_{tu(i)})$$
$$\cup (TR_{th(i)} \ {}_2\&_1 \ TR_{u(i)}) \cup (TR_{th(i)} \ {}_2\&_1 \ TR_{tu(i)})$$

この内，繰り返しにより，

$$TR_{h(i)} \ {}_2\&_1 \ TR_{u(i)}$$

の結果は既に求められているため，

$$(TR_{h(i)} \ {}_2\&_1 \ TR_{tu(i)}) \cup (TR_{th(i)} \ {}_2\&_1 \ TR_{u(i)})$$
$$\cup (TR_{th(i)} \ {}_2\&_1 \ TR_{tu(i)})$$

のみを計算すればよい．（証明終了） ■

アルゴリズム1'と同様に，アルゴリズム2，アルゴリズム3を全解探索用に変更することは簡単である．

6．おわりに

知識ベース処理の1つとして，項関係上の単一化検索演算の繰り返しによる，SLD演繹とSUD演繹の実現

アルゴリズムを提案した．対象とするホーン節の集合が充足不能の場合には，提案したアルゴリズムはそれぞれ反駁を求めて停止することを示した。これにより，ホーン節を知識ベースに格納した場合の，知識ベース処理の論理的根拠を示すことができた．同一の枠組みでさらに単にホーン節に留まらない各種の演繹方法の実現も可能と思われる．

(5) 謝辞

日頃ご指導をいただく内田俊一 ICOT 第4研究室長，林弘 富士通研究所人工知能研究部長，SUD 演繹の証明に助言を頂いた Northwestern 大学の Henschen 教授，有益なコメントを頂いた ICOT の坂井公，佐藤健両氏に感謝します．

(6) 参考文献

1) Gallaire H., Minker J., Nicolas J-M.: Logic and Databases: A Deductive Approach, *Computer Surveys* 16, 2 (1984) .

2) Yokota H., et. al: An Enhanced Inference Mechanism for Generating Relational Algebra Queries, *Proc. of the 3rd ACM SIGACT-ASIGMOD Sympo. on Principles of Database Systems*, pp. 229-238 (1984) .

3) Yokota H., Sakai K., Itoh H.: Deductive Database System based on Unit Resolution, *Proc. of the 2nd Int'l. Conf. on Data Engineering*, pp. 228-235 (1986) .

4) 特集：演繹データベース，情報処理学会誌，Vol. 31, No. 2 (1990).

5) Yokota H., Itoh H.: A Model and an Architecture for a Relational Knowledge Base, *Proc. of 13th Int. Sympo. on Computer Architecture*, pp. 2-9 (1986) .

6) Yokota H., Kitakami H., and Hattori A. : Term Indexing for Retrieval by Unification, *Proc. of 5th Int'l Conf. on Data Engineering*, pp. 313-320 (1989).

7) Apt K.R., van Emden M.H.: Contributions to the Theory of Logic Programming, *J.ACM*, 29, 3 (1982).

8) Lloyd J. W.: *Foundation of Logic Programming*, p. 124, Springer-Verlag (1984).

9) D.W.Loveland: "Automated Theorem Proving", North-Holland, (1978)

10) R.Anderson, W.W.Bledsoe: "A Linear Format for Resolution With Merging and a New Technique for Establishing Completeness", J.ACM, 17, 3 (1970)

(7) 付録

まず，SUD 反駁の健全性を証明する．

【定理５】（SUD反駁の健全性）ホーン節の集合$S = D \cup \{ G \}$ に対してSUD 反駁が存在するならば，S は充足不能である． ■

【証明】 $G = \neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_m \ (1 \leq m)$ とする．SUD 反駁が存在するならば，定義１５より，R_i が空節になる i において，$p_1\theta \in U_i$ ，$p_2\theta \in U_i$ ，…，$p_m\theta \in U_i$ なる θ が存在する． S のモデルを $M(S)$ とすると，$M(S) \supset U_i$ である．よって，$G\theta$ は$M(S)$ において真とは成りえない．つまり，S は充足不能である．（証明終了） ■

次に，SUD反駁の完全性を示す．このため，Andersonら[8] が演繹の完全性を示すために提案した超過リテラルパラメタ（ELP:Excess Literal Parameter）と充足不能最小集合（Minimally Unsatisfiable Set ）を用い，帰納的に証明する．

【定義A1】 S を節の集合$\{C_1,\ldots,C_n\}$とし，LCを$C_1,\ldots,C_n$ 中のリテラル数の合計とする．つまり，

$$LC = \#(C_1) + \ldots + \#(C_n)$$

ここで，$\#(C_i)$ は節C_i 中のリテラル数である．S に対する超過リテラルパラメタを

$$ELP(S) = LC - n$$

とする． ■

【補題A1】 S をC_i が空節でない基底節（変数を持たな

い節）集合とする．$ELP(S)$ が 0 になるのは，全てのC_iが単位節の時でありかつその時だけである．

【証明】自明 ■

【補題A2】 S を，次のような正の単位節 p および節 $\neg p \vee C_2$を含む基底節の集合とする

$$S = \{p, \neg p \vee C_2, C_3,...,C_n\}$$

この時，節$\neg p \vee C_2$をその節とp との導出形C_2 で置き換えた基底節の集合，

$$S' = \{p, C_2, C_3,...,C_n\}$$

は，$ELP(S') = ELP(S) - 1$を満足する．

【証明】自明 ■

【定義A2】節の集合$S = \{C_1,...,C_n\}$が充足不能であり，S からどのようなC_i を除いても充足可能となるようなSを，充足不能最小集合と呼ぶ． ■

【補題A3】S およびS' を補題A2と同様の節集合とする．S が充足不能最小集合の時，以下の何れかが成り立つ．

i) C_2が空節．この場合，$S = \{p, \neg p\}$

ii) C_2は空節でなく，S'が充足不能最小集合

iii) C_2は空節でなく，$S' - \{p\}$ が充足不能最小集合

【証明】もしC_2 が空節だとすると，S の最初の２つの節はp と $\neg p$ であり，この２つの節自身で充足不能最小集合を形成する．よって，$S = \{p, \neg p\}$．次にもしC_2が空節でない場合を考える．S は充足不能であり，S' もやはり充足不能である．S' もしくは$S' - \{p\}$ が充足不能最小集合でないとすると，S'もしくは$S' - \{p\}$ からあ

るC_i $(2 \leq i \leq n)$を除いたものも充足不能となる．$3 \leq i \leq n$ 時充足不能だとすると，C_i はS からS' への導出には使われていないため，$S' - \{C_i\}$も充足不能となり，Sが充足不能最小集合であることに反する．C_2 を除いても充足不能だとすると，$\{p, C_3,...,C_n\}$ が充足不能ということになり，$S = \{p, \neg p \vee C_2, C_3,...,C_n\}$ が充足不能最小集合であることに反する．（証明終了） ■

【補題A4】S が空節を含まない基底ホーン節のみからなる充足不能最小集合の時，S の中に正の単位節pと最左端の負リテラルが$\neg p$ となるような節Cが存在する．

【証明】S の中の全ての正の単位節 p に対して，S 中いかなる節の最左端の負リテラルも $\neg p$ とならないと仮定する．この場合，最左端以外の負リテラルと正の単位節とで導出を行っても新たな単位節は生成されない．つまり，空節は決して生成されない．これはS が充足不能最小集合であるという前提に反する．（証明終了） ■

【補題A5】基底のホーン節からなる集合S が充足不能な時，S において選択関数として最左端の負リテラルを選ぶようなSUD 反駁が存在する．

【証明】S を空節を含まない充足不能最小集合としても，反駁の存在を証明するのに一般性は失われない．

$n = ELP(S)$ の帰納的証明を行う．

$n = 0$ の時：S が充足不能最小集合であることおよび補題A1から，$S = \{p, \neg p\}$．これより反駁は存在する．

$n > 0$ の時：補題A4 から，S 中に正の単位節 p および

$\neg p \vee C$なる節が存在する．S'をpと$\neg p \vee C$をその間の導出形Cで置き換えたものとする．補題A3よりS'もしくは$S' - \{p\}$は充足不能最小集合．補題A2より，$ELP(S') = ELP(S) - 1$よって，帰納法よりSUD反駁が存在する．（証明終了）■

【定理6】（SUD反駁の完全性）ホーン節の集合Sが充足不能な時，Sにおいて選択関数として最左端の負リテラルを選ぶようなSUD反駁が存在する．

【証明】補題A5と持ち上げ定理(Lifting Theorem)[9]より，自明．（証明終了）■

(8) 図

なし

(9) 表

なし

(10) 英文アブストラクト

We have developed resolution algorithms for Horn clauses using retrieval-by-unification (RBU) operations on term relations. A term relation is a table whose elements are well-defined structures called terms. RBU operations are an extension of relational algebra operations using unification to retrieve terms from term relations. SLD and SUD resolution are performed by repeating combinations of RBU operations for term relations consisting of Horn clauses stored as binary trees. The SLD-resolution is a kind of linear resolution with a selection function for definite clauses. The SUD-resolution we propose in this paper is a kind of unit resolution for definite clauses with the same selection function as SLD-resolution. Both SLD and SUD resolutions are sound and complete. If a set of Horn clauses is unsatisfiable, our algorithms obtain the refutation and stop.

Resolution Algorithms for Horn Clauses using Retrieval-by-Unification Operation on Term Relations

Yokota Haruo, Kitakami Hajime, and Hattori Akira

FUJITSU LIMITED