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†*

横田治夫†† 北上始†† 服部彰††

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，RBU 演算を使って 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，RBU 演算による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，探索規則が均等であれば，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．

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)-5)]．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[6)]．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[7)]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 (RBU: Retrieval by Unification) 演算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，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．次に，Prolog 等で行われている SLD 演繹[8),9)]を項関係上の RBU 演算の繰り返しで実現するアルゴリズムを示す．このアルゴリズムは，Prolog と異なり完全であり，ホーン節が充足不能である場合には反駁を求めて停止することを証明する．さらに，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[10)]を，同様に項関係上の RBU 演算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 演繹を実現するアルゴリズムとその停止性を示す．ホーン節中の最左端負リテラルを選択する SUD 演繹は，SLD 演繹と同様，探索規則が均等であれば，健全であり完全である．

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．

† Resolution Algorithms for Horn Clauses Using Retrieval-by-Unification Operation on Term Relations by HARUO YOKOTA, HAJIME KITAKAMI and AKIRA HATTORI (FUJITSU LIMITED).

†† 富士通(株)

* 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．

$^{\forall}n, F_n \cap V = \phi$. ■

【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 $t_1, \cdots, t_n \in T$, $f \in F_n$ $(n \geq 1)$ ならば

$f(t_1, \cdots, t_n) \in T$. ■

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

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

この時次のような tt^m を項タプル (Term Tuple) と呼ぶ．

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

また，t_i を項タプル tt^m の i 番目のアイテムと呼び，$tt^m[i]$ で表す． ■

2.2 単一化検索

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

$v_1, \cdots, v_n \in V$ ($i \neq j$ の時 $v_i \neq v_j$)

$t_1, \cdots, t_n \in T$ $(t_i \neq v_i)$

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 が存在する時，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 とした時，

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

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 $(1 \leq i \leq m)$ と項関係 TR_b^n の j 番目 $(1 \leq j \leq n)$ のアイテム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．

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

$\Leftrightarrow {}^{\exists}tt_a^m \in TR_a^m$, $^{\exists}tt_b^n \in TR_b^n$, $^{\exists}\sigma$,

$tt_a^m[i]\sigma = tt_b^n[j]\sigma$,

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

なお 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 $(1 \leq i \leq m)$ のアイテム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 を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．

$tt_b^m \in TR_b^m$

$\Leftrightarrow {}^{\exists}tt_a^m \in TR_a^m$, $tt_a^m[i] \in V$,

$tt_b^m[k] = tt_a^m[k]$ $(1 \leq k \leq m)$

$tt_c^m \in TR_c^m$

$\Leftrightarrow {}^{\exists}tt_a^m \in TR_a^m$, $tt_a^m[i] \notin V$,

$tt_c^m[k] = tt_a^m[k]$ $(1 \leq k \leq m)$ ■

【定義10】 項関係 TR_a^m から，$a_1, \cdots, a_n$ $(1 \leq a_i \leq m)$ 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，$pr(TR_a^m, [a_1, \cdots, a_n], TR_b^n)$ と略す．

$tt_b^n \in TR_b^n$

$\Leftrightarrow {}^{\exists}tt_a^m \in TR_a^m$,

$tt_b^n[i] = tt_a^m[a_i]$ $(1 \leq i \leq n)$ ■

【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（ただし，変数名の付け替えを行う）． ■

これらの操作を総称して単一化検索 (RBU : Retrieval by Unification) と呼ぶ[6]．このほか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[9]を二進木で表現し，二進木どうしの単一化のみで導出形 (Resolvent) が得られることを示す．

【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 ν 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時，

$\tau(P, B)$

は二進木述語列である．

ただし，ν はいかなる述語中にも現れない変数，τ はいかなる述語中にも現れない2引数関数記号とする． ■

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

$\mathrm{pos}(C)=\tau(p_0,\nu)$

$\mathrm{neg}(C)=\tau(p_1,\tau(p_2,\tau(p_3,\cdots\tau(p_n,\nu)\cdots)))$

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．リテラルが空の場合には，2つの二進木述語列間の共通変数ν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，

$\mathrm{neg}(C)=\nu$

負節（負リテラルのみからなる節）の場合は，

$\mathrm{pos}(C)=\nu$

空節の場合は，

$\mathrm{pos}(C)=\mathrm{neg}(C)=\nu$

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，$\mathrm{neg}(C_1)$ と $\mathrm{pos}(C_2)$ 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 $C_1=p_0\vee\neg p_1\vee\neg p_2\vee\cdots\vee\neg p_n$ 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，

$$p_1\sigma=q_0\sigma \qquad (1)$$

であるような，$C_{2a}=q_0\vee\neg q_1\vee\neg q_2\vee\cdots\vee\neg q_m\ (1\leq m)$ もしくは，$C_{2b}=q_0$ のいずれかである．

（a） C_{2a} の場合

式1から導出形は

$$R_a=(p_0\vee\neg q_1\vee\neg q_2\vee\cdots\vee\neg q_m\vee\neg p_2\vee\cdots\vee\neg p_n)\sigma \qquad (2)$$

となる．C_1 の二進木表現は，

$\mathrm{pos}(C_1)=\tau(p_0,\nu)$

$\mathrm{neg}(C_1)=\tau(p_1,\tau(p_2,\tau(p_3,\cdots\tau(p_n,\nu)\cdots)))$

C_{2a} の二進木表現は，

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

$\mathrm{neg}(C_{2a})=\tau(q_1,\tau(q_2,\tau(q_3,\cdots\tau(q_m,\nu')\cdots)))$

ここで，ν，ν' は定義12から $p_0,\cdots,p_n,q_0,\cdots,q_m$ 中に現れない変数である．式1から，$\mathrm{neg}(C_1)$ と $\mathrm{pos}(C_{2a})$ は単一化可能であり，

$$\tau(p_1,\tau(p_2,\tau(p_3,\cdots\tau(p_n,\nu)\cdots)))\sigma'=\tau(q_0,\nu')\sigma'$$

これより，

$$\sigma'=\{\tau(p_2,\tau(p_3,\cdots\tau(p_n,\nu)\cdots))/\nu'\}\bullet\sigma \qquad (3)$$

これを $\mathrm{pos}(C_1)$ と $\mathrm{neg}(C_{2a})$ に適用すると，

$\mathrm{pos}(C_1)\sigma'$

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

$=\tau(p_0,\nu)\sigma$ （$\because$ ν' は p_0 に含まれないため）

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

$=\tau(q_1,\tau(q_2,\tau(q_3,\cdots\tau(q_m,\nu')\cdots)))\sigma'$

$=\tau(q_1,\tau(q_2,\tau(q_3,\cdots\tau(q_m,\tau(p_2,\tau(p_3,\cdots\tau(p_n,\nu)\cdots)))\sigma$

これらは式2の導出形 R_a の二進木表現にほかならない．

（b） C_{2b} の場合

導出形は，

$$R_b=(p_0\vee\neg p_2\vee\cdots\vee\neg p_n)\sigma \qquad (4)$$

となる．また，C_{2b} の二進木表現は，

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

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

となる．式1から，$\mathrm{neg}(C_1)$ と $\mathrm{pos}(C_{2b})$ は単一化可能であり，$\mathrm{pos}(C_{2b})$ は $\mathrm{pos}(C_{2a})$ 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 $\mathrm{pos}(C_1)$ と $\mathrm{neg}(C_{2b})$ に適用すると，

$\mathrm{pos}(C_1)\sigma'=\tau(p_0,\nu)\sigma'$

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

$\mathrm{neg}(C_{2b})\sigma'=\nu'\sigma'$

$=\tau(p_2,\tau(p_3,\cdots\tau(p_n,\nu)\cdots))\sigma$

これらは，式4の導出形 R_b の二進木表現と同じである（証明終了）． ■

4. 項関係上の RBU 演算による SLD 演繹

SLD 演繹とは，Linear resolution with Selection function for Definite clauses の略[8]で，制限された入力演繹の一種である．Prolog の処理系は，探索規則に制約を加えた SLD 演繹の1つの実現手段とみなすことができる[9]．SLD 演繹では1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，$G_0=G$ とする）の中から選択関数で1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 $C_0,C_1,\cdots$，導出形の列 $G_0,G_1,\cdots$，および代入の列 $\theta_0,\theta_1,\cdots$，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，文献[9]を参照されたい．

次に，ホーン節を二進木表現し1つのホーン節Cを1つの項タプルに対応させて，pos (C) を1番目のアイテム，neg (C) を2番目のアイテムとして，ホーン節の集合を項関係に格納し，その上の RBU 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：確定節の集合 $D(C_k \in D)$ の二進木表現が格納されている2属性の項関係を TR_D とし，
$tt_D \in TR_D$, $tt_D[1]=\text{pos}(C_k)$,
$tt_D[2]=\text{neg}(C_k)$
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
$tt_G \in TR_G$, $tt_G[1]=\text{pos}(G)=\nu$,
$tt_G[2]=\text{neg}(G)$
のように格納する．

Step 2：$uj(TR_G, 2, TR_D, 1, TR_{a\langle 0\rangle})$
$pr(TR_{a\langle 0\rangle}, [2, 4], TR_{b\langle 0\rangle})$
$i=0$

Step 3：もし，$TR_{b\langle i\rangle}=\phi$ ($TR_{b\langle i\rangle}$ 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：$vr(TR_{b\langle i\rangle}, 2, TR_{c\langle i\rangle}, TR_{a\langle i\rangle})$

Step 5：もし，$TR_{c\langle i\rangle} \neq \phi$ なら，反駁に成功して終了，$TR_{c\langle i\rangle}$ 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：$i=i+1$
$uj(TR_{a\langle i-1\rangle}, 2, TR_D, 1, TR_{a\langle i\rangle})$
$pr(TR_{a\langle i\rangle}, [1, 4], TR_{b\langle i\rangle})$
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 $uj(TR_G, 2, TR_D, 1, TR_{a\langle 0\rangle})$ と，定義8から，

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

これより，
$\text{neg}(G)\theta_0=tt_G[2]\theta_0=tt_D[1]\theta_0=\text{pos}(C_0)\theta_0$
この時，定理1と定義14より，
$\text{neg}(C_0)\theta_0=\text{neg}(G_1)$
となる．次に，$pr(TR_{a\langle 0\rangle}, [2, 4], TR_{b\langle 0\rangle})$ から，
$tt_{b\langle 0\rangle}[1]=tt_{a\langle 0\rangle}[2]=tt_G[2]\theta_0=\text{neg}(G)\theta_0$
$tt_{b\langle 0\rangle}[2]=tt_{a\langle 0\rangle}[4]=tt_D[2]\theta_0=\text{neg}(C_0)\theta_0$
$=\text{neg}(G_1)$
つまり，$TR_{b\langle 0\rangle}$ の第一属性は最初の負節中の変数に対する代入を，$TR_{b\langle 0\rangle}$ の第二属性は一番目の導出形の負リテラルを表している．次に，Step 4 の $vr(TR_{b\langle i\rangle}, 2, TR_{c\langle i\rangle}, TR_{a\langle i\rangle})$ と，定義9から，

$tt_{c\langle 0\rangle} \in TR_{c\langle 0\rangle}$
$\Leftrightarrow {}^{\exists}tt_{b\langle 0\rangle} \in TR_{b\langle 0\rangle}$,
$tt_{c\langle 0\rangle}[2]=tt_{b\langle 0\rangle}[2]=\text{neg}(G_1) \in V$,
$tt_{c\langle 0\rangle}[1]=tt_{b\langle 0\rangle}[1]=\text{neg}(G)\theta_0$
$tt_{a\langle 0\rangle} \in TR_{a\langle 0\rangle}$
$\Leftrightarrow {}^{\exists}tt_{b\langle 0\rangle} \in TR_{b\langle 0\rangle}$,
$tt_{a\langle 0\rangle}[2]=tt_{b\langle 0\rangle}[2]=\text{neg}(G_1) \notin V$,
$tt_{a\langle 0\rangle}[1]=tt_{b\langle 0\rangle}[1]=\text{neg}(G)\theta_0$

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

$tt_{b\langle 1\rangle}[1]=\text{neg}(G)\theta_0 \bullet \theta_1$
$tt_{b\langle 1\rangle}[2]=\text{neg}(C_1)\theta_1=\text{neg}(G_2)$
ただし，
$\text{neg}(G_1)\theta_1=\text{pos}(C_1)\theta_1$
さらに，これを繰り返して，
$tt_{b\langle i\rangle}[1]=\text{neg}(G)\theta_0 \bullet \cdots \bullet \theta_i$
$tt_{b\langle i\rangle}[2]=\text{neg}(C_i)\theta_i=\text{neg}(G_{i+1})$
ただし，
$\text{neg}(G_i)\theta_i=\text{pos}(C_i)\theta_i$
となる．定義14と定義13から，G_i が空節，つまり $\text{pos}(G_i)=\text{neg}(G_i)=\nu$ の時 SLD 反駁が得られたことになる．常に $\text{pos}(G_i)=\nu$ であるため，$\text{neg}(G_i)=tt_{b\langle i-1\rangle}[2] \in V$ つまり，繰り返し中に $TR_{c\langle i\rangle}$ が ϕ でなくなる時，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 $uj(TR_{a\langle i-1\rangle}, 2, TR_D, 1, TR_{a\langle i\rangle})$ で，前段の導出形と導出可能なすべ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，節の宣言された順番に従うため，完全ではない[9]．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すべての候補が適用されるため，探索規則が均等となり，$D \cup \{G\}$ が充足不能である場合は必ず SLD 反駁が得られる．

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

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

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

$tt_G \in TR_G$, $tt_G[1]=\mathrm{pos}(G)=\nu$,
$tt_G[2]=\mathrm{neg}(G)$

のように格納する．

Step 2：$TR_{x(0)}=\phi$
$uj(TR_G, 2, TR_D, 1, TR_{a(0)})$
$pr(TR_{a(0)}, [2,4], TR_{b(0)})$
$i=0$

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，$TR_{d(i)}=\phi$（$TR_{d(i)}$ 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，Step 5 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 演繹においてすべ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′ はそのすべ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′ では可能なすべ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すべ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．すべての場合の反駁が有限時間内に得られる場合には，$TR_{d(n)}=\phi$ なる n が存在することになる．この時，可能なすべての反駁によって得られる代入は，$TR_{c(i)}$ $(1 \leq i \leq n)$ の第一属性に現れる．$TR_{x(0)}=\phi$ 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 RBU 演算による 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[10]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 RBU 演算を用いた 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 $(G \cup (D-U_0)=H_0)$ 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 $U_0, U_1, \cdots$，複合節の集合の列 $H_0, H_1, \cdots$，および代入の列 $\theta_0, \theta_1, \cdots$，から得られる反駁を SUD 反駁と呼ぶ． ■

【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 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】 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,
$tt_{u(0)} \in TR_{u(0)}$,
$tt_{u(0)}[1] = \mathrm{pos}(u_k)$,
$tt_{u(0)}[2] = tt_{u(0)}[3] = \mathrm{neg}(u_k) = \nu$
また, 複合節の集合 ($C_h \in H$) の格納されている3属性の項関係を $TR_{h(0)}$ とし,
$tt_{h(0)} \in TR_{h(0)}$,
$tt_{h(0)}[1] = \mathrm{pos}(C_h)$,
$tt_{h(0)}[2] = tt_{h(0)}[3] = \mathrm{neg}(C_h)$
とする.

Step 2: $i = 0$

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

Step 4: $vr(TR_{tu(i)}, 1, TR_{c(i)}, TR_{d(i)})$
もし, $TR_{c(i)} \neq \phi$ 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 Step 3 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,

$$
\begin{aligned}
& tt_{b(i)} \in TR_{b(i)} \\
& \Leftrightarrow {}^{\exists} tt_{h(i)} \in TR_{h(i)},\ {}^{\exists} tt_{u(i)} \in TR_{u(i)},\ {}^{\exists}\theta_i, \\
& \quad tt_{h(i)}[2]\theta_i = tt_{u(i)}[1]\theta_i, \\
& \quad tt_{b(i)}[1] = tt_{h(i)}[1]\theta_i = \mathrm{pos}(C_i)\theta_i \\
& \qquad = \mathrm{pos}(R_i) \\
& \quad tt_{b(i)}[2] = tt_{u(i)}[2]\theta_i = \mathrm{neg}(u_i)\theta_i \\
& \qquad = \mathrm{neg}(R_i) \\
& \quad tt_{b(i)}[3] = tt_{h(i)}[3]\theta_i = \mathrm{neg}(C_0)\theta_0 \bullet \cdots \bullet \theta_i
\end{aligned}
$$

この時, $tt_{h(i)}[2]\theta_i = tt_{u(i)}[1]\theta_i$ から, $\mathrm{neg}(C_i)\theta_i = \mathrm{pos}(u_i)\theta_i$ となる. また, 定義13より, R_i が単位節なら $\mathrm{neg}(R_i)$ 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 ϕ 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_i が空節となる. これは,

$$\mathrm{pos}(R_i) = \mathrm{neg}(R_i) = \nu$$

つまり,

$$tt_{tu(i)}[1] \in V$$

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 ϕ でなくなる. その時, 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,
$tt_{u(0)} \in TR_{u(0)}$,
$tt_{u(0)}[1] = \mathrm{pos}(u_k)$,
$tt_{u(0)}[2] = tt_{u(0)}[3] = \mathrm{neg}(u_k) = \nu$
また, 複合節の集合 ($C_h \in H$) の格納されている3属性の項関係を $TR_{h(0)}$ とし,
$tt_{h(0)} \in TR_{h(0)}$,
$tt_{h(0)}[1] = \mathrm{pos}(C_h)$,
$tt_{h(0)}[2] = tt_{h(0)}[3] = \mathrm{neg}(C_h)$
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，$TR_{c(i)} \neq \phi$ 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 : $uj(TR_{h(i)}, 2, TR_{tu(i)}, 1, TR_{ax(i)})$,
$pr(TR_{ax(i)}, [1, 5, 3], TR_{bx(i)})$,
$vr(TR_{bx(i)}, 2, TR_{ux(i)}, TR_{hx(i)})$,
$uj(TR_{th(i)}, 2, TR_{u(i)}, 1, TR_{ay(i)})$,
$pr(TR_{ay(i)}, [1, 5, 3], TR_{by(i)})$,
$vr(TR_{by(i)}, 2, TR_{uy(i)}, TR_{hy(i)})$,
$uj(TR_{th(i)}, 2, TR_{tu(i)}, 1, TR_{az(i)})$,
$pr(TR_{az(i)}, [1, 5, 3], TR_{bz(i)})$,
$vr(TR_{bz(i)}, 2, TR_{uz(i)}, TR_{hz(i)})$,

Step 6 : $un(TR_{u(i)}, TR_{tu(i)}, TR_{u(i+1)})$,
$un(TR_{h(i)}, TR_{th(i)}, TR_{h(i+1)})$,
$un(TR_{ux(i)}, TR_{uy(i)}, TR_{uw(i)})$,
$un(TR_{uw(i)}, TR_{uz(i)}, TR_{tu(i+1)})$,
$un(TR_{hx(i)}, TR_{hy(i)}, TR_{hw(i)})$,
$un(TR_{hw(i)}, TR_{hz(i)}, TR_{th(i+1)})$

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8】 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，

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

このうち，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3で用いた最適化手法は，演繹データベースのボトムアップ計算におけるセミナイーブ法[2]と同質のアプローチである．演繹データベースでは，ほかにも，問い合わせ内容を利用した最適化手法であるマジックセット法[2),11)]なども提案されており，RBU 演算による SUD 演繹にも同様な手法が適用できるものと思われる（ただし，演繹データベースと異なり，関数記号を扱うため，そのことを考慮する必要がある）．また，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容易である．

6. おわりに

知識ベース処理の1つとして，項関係上の RBU 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．これにより，ホーン節を知識ベースに格納した場合の，演繹処理の論理的根拠を示すことができた．つまり，単位節や関数記号の有無にこだわらない一般のホーン節に対して，Prolog 等の論理プログラミング言語処理系とは異なるデータベース的アプローチによって，集合としての管理を考慮し，量が増大した場合にも対応できる知識ベースの枠組みを提供することが可能となった．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．

謝辞 日頃ご指導をいただく内田俊一 ICOT 研究部長，伊藤英則名古屋工大教授，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いただいた Northwestern 大学の Henschen 教授，有益なコメントをいただいた ICOT の坂井公，佐藤健両氏に感謝します．

参考文献

1) Gallaire, H., Minker, J. and Nicolas, J-M.: Logic and Databases: A Deductive Approach, *Comput. Surv.*, Vol. 16, No. 2, pp. 153-185 (1984).
2) Bancilhon, F. and Ramakrishnan, R.: An Amateur's Introduction to Recursive Query Processing Strategies, *Proc. of the ACM SIGMOD '86*, pp. 16-52 (1986).
3) Yokota, H., Kunifuji, S., Kakuta, T., Miyazaki, N., Shibayama, S. and Murakami, K.: An Enhanced Inference Mechanism for Generating

Relational Algebra Queries, *Proc. of the 3rd Symp. on Principles of Database Systems*, pp. 229-238 (1984).
4) Yokota, H., Sakai, K. and Itoh, H.: Deductive Database System Based on Unit Resolution, *Proc. of the 2nd Int'l. Conf. on Data Engineering*, pp. 228-235 (1986).
5) 特集：演繹データベース，情報処理，Vol. 31, No. 2 (1990).
6) Yokota, H. and Itoh, H.: A Model and an Architecture for a Relational Knowledge Base, *Proc. of the 13th Int'l. Symp. on Computer Architecture*, pp. 2-9 (1986).
7) Yokota, H., Kitakami, H. and Hattori, A.: Term Indexing for Retrieval by Unification, *Proc. of the 5th Int'l. Conf. on Data Engineering*, pp. 313-320 (1989).
8) Apt, K. R. and van Emden, M. H.: Contributions to the Theory of Logic Programming, *J. ACM*, Vol. 29, No. 3, pp. 841-862 (1982).
9) Lloyd, J. W.: *Foundation of Logic Programming*, p. 124, Springer-Verlag (1984).
10) Loveland, D. W.: *Automated Theorem Proving*, p. 405, North-Holland (1978).
11) Bancilhon, F., Maier, D., Sagiv, Y. and Ullman, J. D.: Magic Sets and Other Strange Ways to Implement Logic Programs, *Proc. of 5th Symp. on Principles of Database Systems*, pp. 1-15 (1986).
12) Anderson, R. and Bledsoe, W. W.: A Linear Format for Resolution with Merging and a New Technique for Establishing Completeness, *J. ACM*, Vol. 17, No. 3, pp. 525-534 (1970).

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】 $G=\neg p_1\vee\neg p_2\vee\cdots\vee\neg p_m$ $(1\leq m)$ 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，$p_1\theta\in U_i, p_2\theta\in U_i, \cdots, p_m\theta\in U_i$ 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[12] 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 $\{C_1, \cdots, C_n\}$ とし，LC を $C_1, \cdots, C_n$ 中のリテラル数の合計とする．つまり，

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

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，すべての C_i が単位節の時でありかつその時だけである．

【証明】 LC と n とが等しいことから明らか． ■

【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．

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

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，

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

は，$ELP(S')=ELP(S)-1$ を満足する．

【証明】 n が変わらずに LC が1減ることから明らか． ■

【定義A2】 節の集合 $S=\{C_1, \cdots, C_n\}$ 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，以下のいずれかが成り立つ．

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，$\{p, C_3, \cdots, C_n\}$ が充足不能ということになり，$S=\{p,$

$\neg p \vee C_2, C_3, \cdots, C_n\}$ 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の中のすべての正の単位節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 反繹が存在する．

【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）．■

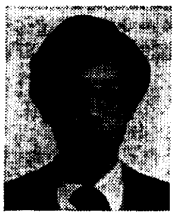
【定理6】（SUD 反駁の完全性）ホーン節の集合Sが充足不能な時，Sにおいて選択関数として最左端の負リテラルを選ぶような SUD 反駁が存在する．

【証明】 補題 A5 と持ち上げ定理（Lifting Theorem）[10]より，直ちに示される（証明終了）．■

（平成2年4月 4 日受付）
（平成2年7月10日採録）

横田 治夫（正会員）

1957 年生．1980 年東京工業大学工学部電子物理工学科卒業．1982年同大学院情報工学専攻修士課程修了．同年4月，富士通(株)入社．同年6月より(財)新世代コンピュータ技術開発機構に出向．1986年4月(株)富士通研究所に帰属．データベースマシン，演繹データベース，知識ベースシステム，並列記号処理の研究・開発に従事．電子情報通信学会，人工知能学会各会員．



北上 始（正会員）

1952 年生．1976 年東北大学大学院修士課程修了．同年富士通(株)入社．1978 年～82 年(株)富士通研究所において関係データベース管理システムの研究・開発に従事．1982年6月～85年5月(財)新世代コンピュータ技術開発機構において知識ベース管理システムの研究に従事．1985年6月(株)富士通研究所に帰属．静止衛星の姿勢制御ソフトウェアの開発，関係データベース管理システムの研究・開発，知識ベース管理システムの研究．日本認知学会，人工知能学会各会員．

服部 彰（正会員）

昭和24年生．昭和47年大阪大学工学部電子工学科卒業．昭和 49 年同大学院工学研究科修士課程修了．同年(株)富士通研究所入社．階層メモリ，記号処理マシンの研究を経て，並列コンピュータの研究開発に従事．現在，人工知能研究部第3研究室長．電子情報通信学会，人工知能学会各会員．