TM-0291

非正規型モデルへの
演繹的アプローチ

横田 一正

March, 1987

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

非正規形モデルへの演繹的アプローチ

横田一正
（財）新世代コンピュータ技術開発機構

関係モデルを一階理論の一部とし，データベースを証明論的に考える演繹データベースのアプローチは，データベース拡張の研究に大きく役立ってきた．本稿では，非正規形モデルのレコードを表現する，代数構造をもった（一階述語項の拡張としての）項を定義し，それに基づいた論理型言語によって，非正規型モデルに対する演繹的アプローチを試みる．これは現在ICOTで研究開発中の知識ベース管理システムKappaの形式的枠組でもある．

Deductive Approach for Unnormalized Relations

Kazumasa YOKOTA

Institute for New Generation Computer Technology
1-4-28, Mita, Minato-ku, Tokyo 108

A deductive database, in which a relational model is considered as a part of the first order theory, plays an important role for extention of database theory. In this paper, we define an extended term, by which an unnormalized record can be represented, and we take a deductive approach for unnormalized relations in the framework of a logic programming language based on the term. This is also a formal framework of KAPPA, which is a knowledge base management system under development at ICOT.

1. はじめに

関係モデルは数学的意味が明確であるため，単に1つのデータ・モデルとしてではなく，データベース理論全体の発達に大きな役割を果してきた．そしてデータベース理論の拡張のために，関係モデルと一階述語論理との関係が多く研究されている〔Gallaire 78〕．その1つに演繹データベースがある〔Gallaire 84, Reiter 84〕．関係モデルの伝統的な形式化が関係を解釈と考える，いわばモデル論的なアプローチであったのに比べると，この演繹データベースは関係を一階理論の一部と置き，定理証明系としてデータベース操作を考える，いわば証明論的アプローチである．この枠組によって，否定や選言的データの導入，あるいはデータ，推論規則および一貫性制約の統一的記述の問題などが，一階理論の枠組の中で研究されている．

ところが関係モデルを実際の応用面から考えると，エンジニアリング・データベースやオフィス・データベースなど，その第一正規形の条件のふさわしくないものが多い．非正規形モデルはその解決のための1つとして研究されている．ICOTでは逐次型推論マシン(PSI)上にさまざまな知識情報処理システムの開発を計画しているが，そのためにさまざまな構造をもったデータや知識を扱うためのデータベース／知識ベース管理システムが必要とされている．その実現を目指すシステムKappaは，効率上の問題と構造の扱いから関係モデルではなく，非正規形モデルを内部モデルとしている〔Yokota 86,87a〕．これまで非正規形モデルの研究は数多くなされているが〔Makinouchi 77, Schek 82, Fischer 83, Kambayashi 83, Abitchoul 84, Pistor 86〕，関係モデルと異なり，このモデルを一階述語論理の範囲内で考えることが難しいため，論理と関連づける試みはわずかしかない．異なった視点から，関係モデル，階層モデルおよび網形モデルを共通の形式的枠組で統合しようとするデータベース論理〔Jacobs 82〕が提案されているが，それを非正規形モデルの形式的枠組にするには不充分である．しかし最近，他の視点から非正規形のレコード構造に対応する論理の提案がなされ始めた．自然言語諸分野の統一的記述を目指すCIL〔Mukai 86〕，型の継承を扱うKBL〔Ait-Kaci 84〕やそれをPrologに組み込んだLOGIN〔Ait-Kaci 86〕あるいは対象指向データベースの形式化を目指したO-論理〔Maier 86〕，などである．

Kappaはデータベース管理システムと知識ベース管理システムの両面をもったシステムを目指している．そのためには，非正規形モデルを直接表現すると共に，さまざまな知識表現言語の内部表現ともなる形式的枠組が必要である．本稿では，そのようなKappaの基礎理論として現在検討している形式的枠組を提案する．これは上の新しい枠組の1つとして位置づけることができる．すなわち属性と値の対の集合として表現されるレコードを基にした演繹である．まず次節では，例を中心にして概要を説明する．そして3節では非正規形データベースの演繹的アプローチの全体を概説し，4節で，Kappaの形式的な取組みの紹介をし，最後に今後の課題を述べる．本稿は，動機と全体的な枠組の紹介であり，詳細は〔Yokota 87b〕を参照されたい．

2. レコードと演繹の例

『レコード』は実世界の情報を表現するためのもっとも基本的な表現形式の1つである．レコードを単に均質の構造をもった属性値の順序集合の集合と考えると，表現能力の点で多くの問題が生じてくる．しかしレコードは，その表現方法を変えるだけで，部分情報の扱いやタクソノミーの扱いなどで，知識表現という面で重要な位置を占めており，そこでさまざまなレコード表現が提案されている．この節では例を中心にして，Kappaで考えているレコードの直感的な説明する．まず単純な例から始めよう．

〔例1〕名前，年齢，出身地からなるレコード

名前'"金枝上"＊年齢'28＊出身地'"神戸"，
名前'"河村"＊年齢'26＊出身地'"名古屋"

属性名と値の('''による)組の('＊'による)組合せがレコードとなる．値は属性名によって修飾されているので，位置はどこにあってもよい．

〔例2〕名前と趣味からなるレコード

名前'"金枝上"＊趣味'{"スキー","テニス","読書"}，
名前'"河村"＊趣味'{"音楽鑑賞","スキー"}

複数の値をもつ場合は集合として表現できる．したがって名前の属性値も単一要素からなる集合(singleton)と考える．このとき括弧は省略することが多い．集合表現されたレコードはそれを分解(行アンネスト)したものと同じ意味をもっている．つまり2番目のレコードは

名前'"河村"＊趣味'"音楽鑑賞"＊趣味'"スキー"

または

名前'"河村"＊趣味'"音楽鑑賞"，
名前'"河村"＊趣味'"スキー"

とすることができる．

〔例3〕階層構造をもったレコード

学生'(名前'(姓'"斉藤"＊名'ω)＊所属'X
　＊趣味'{"野球","音楽"}
　＊出身校'(学校名'"三田"＊所在地'"東京")),

このように属性の階層構造も表現することができる．"ω"は値が不明なことを表わしている．なお本稿ではレコード値の集合については本稿では詳しく触れない．

〔例4〕親子関係のレコード

親'{"孝","桜"}＊子供'{"明","歯","優"}，
親'{"明","薫"}＊子供'{"純"}，

親`{"徹","麗"}＊子供`φ，
子供`{"孝"}，
親`{"勇","夏"}＊子供`{"桜","博"}
3番目のレコードは子供がいないことを表わしている．そして4番目のレコードは親がわからないことを表わしている．このように空値を，"存在しない"と"不明"の2種類で表現する．

〔例5〕親子関係への質問
通常のPrologと同じく，例4の各レコードを単位節(ファクト)と考え，質問をゴールとして与える．
・"明"の親を求める：←Y∋"明"|親`X＊子供`Y．
・"明"と"薫"の共通の親を求める：
←Y⊇{"明","薫"}|親`X＊子供`Y．
・子供のいない親を求める：←Y＝φ|親`X＊子供`Y．
・親の不明の子供を求める：←X＝ω|親`X＊子供`Y．
答えはそれぞれ，X＝{"孝","桜"}，X＝{"孝","桜"}，X＝{"徹","麗"}，Y＝{"孝"}が返される．この質問の特徴は，集合値のほかに空値で質問することができること，また結果が集合として返ってくることである．"|"の前にある条件は"|"の後の対応する変数が具象化されたときの制約である．単一化については4.2節で触れる．

〔例6〕再帰型の質問
Prolog同様に，先祖関係はレコードを使ったプログラム節として書くことができる：
先祖`X＊子孫`Y ← 親`X＊子供`Y．
先祖`X＊子孫`Y ← 親`Z＊子供`Y，先祖`X＊子孫`Z．
すると"純"の先祖を求める質問は次のようになる：
←Y∋"純"|先祖`X＊子孫`Y．
答えはXに，{"明","薫"}，{"孝","桜"}，{"勇","夏"}が返される．このようにレコード構造に基づくプログラムを自由に書くことによって複雑な質問を処理することができる．

〔例7〕レコードを使った論理式
・ ∀X,∃Y．人`(名前`X＊趣味`Y)．
・ ∀M,∀D．町`(町長`M＊町名`D)
←町`(町民`M＊町名`D)
∨町`(町民`M＊町名`"特別町")．
このようにして一階述語論理式はレコードに基づいた論理式として表現できる．

本稿では，上記のようなレコード表現と，それに基づいた演繹の形式的枠組を提案する．非正規形モデルの枠組としては，レコード値の集合にまだ触れていないので不充分であるが，拡張可能性をもっている．また階層で無限木を表現することによって，タクソノミーの表現も可能になる．

3．非正規形モデルと演繹データベース

関係モデルを基にした演繹データベースでは，関係モデルと一階述語論理の親和性が大きな役割を果している．つまりタプルは1つの述語，Prologの単位節として表現できるので，Prolog自体を関係モデルの拡張と考えることができる．そこで関係モデルの拡張（否定や選言的情報の導入や推論機構の付加など）を一階理論の枠内で研究することができる．

ところが非正規形モデルの場合，レコード(タプル)を1つの述語に対応させることは難しい．なぜならば属性間の階層関係を関数のネストに対応させ，集合値をもつ形に拡張しても，非正規形モデル独特のネスト／アンネストを表現することができないからである．そこでKappa では非正規形モデルのレコード表現のための（一階述語項の拡張としての）項を定義し，その項に基づいた論理型言語を定義することにした．そしてその言語の上で，レコード，推論規則，一貫性制約を記述し，データベース操作を定理証明系に置き換えるだけでなく，再帰型を始めとする複雑な質問も扱うことにした．さらに空値の扱いも考慮した．

さらにこの言語は，単に非正規形モデルに対する演繹的アプローチというにとどまらず，多くの拡張性をもっている．そのためにレコードの一般的な定義を考えてみよう．属性集合をA，定数集合をC，そのベキ集合をΔ，A の上の木領域をT，その葉集合をleaf(T)とすると，種々のレコードは以下の関数として定義できる：
① 平坦なレコード： A→C，
② 階層構造をもったレコード： leaf(T)→C，
③ 集合値をもつ階層構造のレコード： leaf(T)→Δ．
関係モデルでは，タプルは属性集合から値の集合への関数として定式化される．これに対し本稿で扱うのは，③の，属性は限定された階層構造をもち，集合値をもつレコード構造である．これからさまざまな拡張が考えられる．

〔Ait-Kaci 84,86〕は，定義域を葉集合から木領域全体に拡張しノード間にタグを付けることによって，'ψ-項'と呼ぶ項でタクソノミーを表現している．〔Ait-Kaci 84〕ではそれを項書換えに基づく言語で，〔Ait-Kaci 86〕ではそれをPrologに埋め込んで『継承』を自然に扱っている．本稿の言語にも，このような拡張性を考えることができる．〔Maier 86〕は，Ait-Kaciと同様の形式化によって'O-項'を定義し，対象指向データベースと論理型言語の融合をおこなっている．これは'型'と'対象'の類似によっている．しかし彼が採用しているのはモデル論的アプローチであり，〔Reiter 84〕で指摘されてる問題点をそのまま残している．〔Mukai 86〕は，データベース的な観点をまったくもっていないが，本稿のアプローチにもっとも近いものである．これは単にデータベースにとどまらず，他に諸分野とも共通に使用できる可能性を示している．

4. Kappaの形式的枠組

4.1 項と部分タグ木

まず定数集合Cとそのベキ集合Δ，空属性eを含む属性集合Aの存在を仮定する．A は接合演算"^"でモノイドとなっている．この接合属性の集合をA*で表わす．さらに変数集合V と未定義定数の集合Ω＝{ω，o}の存在を仮定する．

〔定義〕項

i）a^X，ただしa∈A,X∈Δ∪V∪{ω}，

ii）a1^t1*…*an^tn，ただしai∈A,そしてtiは項である，

iii）a1^t1+…+an^tn，ただしai∈A,そしてtiは項である，

iv）以上によって作られるものだけが項である．

〔定義〕項の演算

属性接合子"^"と項構成子'*','+'について以下の演算を定義する：

(1) $a1^{\wedge}(a2^{\wedge}X)=(a1^{\wedge}a2)^{\wedge}X$,

(2) $a0^{\wedge}(a1^{\wedge}t1*\cdots*an^{\wedge}tn)=a0^{\wedge}(a1^{\wedge}t1)*\cdots*a0^{\wedge}(an^{\wedge}tn)$,

(3) $a0^{\wedge}(a1^{\wedge}t1+\cdots+an^{\wedge}tn)=a0^{\wedge}(a1^{\wedge}t1)+\cdots+a0^{\wedge}(an^{\wedge}tn)$.

また項間の演算'*'と'+'については以下を定義する：

(4) $t1*t2=t2*t1$ *_交換

(5) $t1*(t2*t3)=(t1*t2)*t3$ *_結合

(6) $t1+t2=t2+t1$ +_交換

(7) $t1+(t2+t3)=(t1+t2)+t3$ +_結合

(8) $t+t=t$ +_ベキ等

(9) $t1*(t2+t3)=(t1*t2)+(t1*t3)$ *,+_分配

〔定理〕任意の項tは以下の形式(単項標準形)に変換できる：

$$t1+\cdots+ti+\cdots+tn$$
$$=(ti1*\cdots*tip)+\cdots+(ti1*\cdots*tir)+\cdots+(tn1*\cdots*tnq),$$

ただしtij＝aij^X (aij∈A*，X∈Δ∪V∪Ω)である．この単項標準形の各ti(1≦i≦n)を単項という．そして

a_set(ti)＝{ti1,…,tir},

set(t)＝{a_set(t1),…,a_set(ti),…,a_set(tn)},

t_set(t)＝{t1,…,tn}

で表わす．

〔定義〕整項と両立性

単項t＝a1^X1*…*an^Xnが，任意のi,jに対して

i）i≠jならばai≠aj，

ii）ai＝ai1^a(またはai＝ai1^a^ai2)かつa≠eのとき，aj＝aj1^a(あるいはaj＝aj1^a^aj2)が存在するならばai1＝aj1

のとき整項という．２つの整項tとuが両立するとは，任意のai^X＝ai1^a^X(またはai^X＝ai1^a^ai2^X) ∈a_set(t)に対して，aj^Y＝aj1^a^Y (またはaj^Y＝aj1^a^aj2^Y)∈a_set(u)が存在するならばai1＝aj1であることをいう．

そして項に対応した意味領域を定義する．まずA*の部分集合で，接頭閉包(prefix closed)の条件を満たすものを木という．そして木Tの葉の集合をleaf(T)で表わす．

〔定義〕部分タグ木(PTT)

木Tと，leaf(T)からC∪{o}への部分関数Fの対(T,F)を部分タグ木という．また属性a∈A とPTT: PT＝(T,{(a1,c1),…,(an,cn)})に対して，a^PT＝(a^T,{(a^a1,c1),…,(a^an,cn)})と定義する．またPTTのサブクラスで接合属性が整項と同じ条件をもつものを整木といい，そして整木間の両立性も同様に定義できる．また両立する整木の集合を整木集合といい，整木集合間の両立性も定義する．

〔定義〕整木の順序と演算

２つの両立する整木WT1＝(T1,F1)とWT2＝(T2,F2)の間に順序≦，交わり操作∧，結合操作∨を定義する：

WT1≦WT2 ⇔ F1⊆F2,

WT1i∧WT2j＝(T1i∩T2i,F1i∩F2j),

WT1i∨WT2j＝(T1i∪T2i,F1i∪F2j).

そしてそれを整木集合ST1＝{WT11,…,WT1m} (WT1j＝(T1i,F1i))，ST2＝{WT21,…,WT2n} (WT2j＝(T2j,F2)) の間にも拡張する：

ST1≦ST2 ⇔ ∀WT1i∈ST1.∃WT2j∈ST2.WT1i≦WT2j.

ST1∧ST2＝{WT11∧WT12,…,WT11∧WT2n,
…,WT1m∧WT21,…,WT1m∧WT2n},

ST1∨ST2＝{WT11∨WT12,…,WT11∨WT2n,
…,WT1m∨WT21,…,WT1m∨WT2n}.

次に整項と整木集合を関係づける．そのためにまず割当てηを定義する：

i）c ∈C のとき，η[c]＝c，

ii）φ∈Δのとき，η[φ]＝o，

iii）x ∈V のとき，η[x]∈Δ，

iv）ω∈Ωのとき，η[ω]は定義されない．

そして整項の整木への意味関数ιを定義する：

i）t＝a^φ(a∈A,φ∈Δ)のとき，

ι[t]＝({e,a},{(a,o)}),

ii）t＝a^s (a∈A,s＝{c1,…,cn}∈Δ,n≠0)のとき，

ι[t]＝
{({e,a},{(a,η[c1])}),…,({e,a},{(a,η[c1])})},

iii）t＝a^x (a∈A,x∈V)のとき，ι[t]＝ι[a^η[x]]，

iv）t＝a^ω(a∈A,ω∈Ω)のとき，

ι[t]＝({{e,a},φ)},

v）t＝a^t1(a∈A)のとき，ι[t]＝a^ι[t1]，

vi）tが整項a1^t1*…*an^tnのとき，

ι[t]＝(ι[a1^t1])∨…∨(ι[an^tn]).

〔定理〕 任意の整項は変数割当てηが与えられたとき整木

集合に対応させることができ，逆に任意の整木1つからなる集合は変数をもたない整項として表現できる．また項の上で定義した演算('+'を含まない演算）は両立する整木の集合で成り立つ．

4.2 単一化

変数への代入θ＝{(x1,t1),…,(xn,tn)}は通常通り定義できるので，それを項に拡張しよう：

〔定義〕項への代入

i) t＝a^s (s∈Δ∪Ω)ならば，　θ(t)＝a^s，

ii) t＝a^x (x∈V)ならば，　θ(t)＝a^θ(x)，

iii) t＝a1^t1*…*an^tnならば，

θ(t)＝a1^θ(t1)*…*an^θ(tn)，

iv) t＝a1^t1+…+an^tnならば，

θ(t)＝a1^θ(t1)+…+an^θ(tn)．

項tに代入θをおこなう場合，tθとも書く．代入の複合は通常通り定義できる．それをtθφと書く．

項の集合の上で，改名代入に関する同値関係，単項標準形で結合律，交換律に関する同値関係，および'+'のベキ等に関する同値関係の推移閉包から作られる同値関係を単に'＝'で表わす．そして一般性を損うことなく，この同値類の代表元だけを使用する．

〔補題〕両立する整項tとuはそれぞれ

t'＝(a1^t1)*…*(ak^tk)*(b1^X1)*…*(bn^Xn)，

u'＝(a1^u1)*…*(ak^uk)*(c1^Y1)*…*(cm^Ym)

と変形することができる．ただしt1,…,tk，u1,…,ukは項で，すべての1≦ai≦akについてtiまたはuiのどちらかがe^Zi(∈Δ∪V∪Ω)の形をしている．そしてX1,…,Xn,Y1,…,Ym∈Δ∪V∪Ωであり，任意の1≦bi≦bn，1≦cj≦cmに対してbi≠cjである．このときtとuに対して，t"＝t*c1^ω*…*cm^ω，u"＝u*b1^ω*…*bn^ωをそれぞれt,uの両立化項という．

〔定義〕整項の単一化

両立する整項tとuに対しそれらの単一化を定義する．それぞれの両立化項を，t'＝a1^t1*…*an^tn，u'＝a1^u1*…*an^unと仮定する．このとき代入θが存在して

① ti＝ui＝φの場合，　si＝φ，

② ti,ui∈Δ-φの場合，　si＝ti∩ui≠φ，

③ tiまたはuiが変数の場合，　si＝xiθ＝yiθ，

④ ①,②,③以外のとき，　siは定義されない

が成り立つとき，単一化項(a1'^s1')*…*(am'^sm')が定義される．このときθをtとuの単一子(unifier)という．②または③が成り立たないとき，単一化は失敗するという．また mguは通常通り定義でき，tとuのmguをmgu(t,u)で表わす．そしてθがmguのときtθ＝uθをuni(t,u)で表わす．

〔定義〕整項の一般化

整項tとuに対しそれらの一般化を定義する．tとuの両立化項に含まれる変数の集合をV'とし，V'×V'からVへの単射をそれぞれρとし，そして一般化アルゴリズムγを次のように決める：

i) si,sj∈Δのとき，　γ(si,sj)＝si∪sj，

ii) ω,∈Ωのとき，　γ(ω,ω)＝ω，

iii) tが項でω∈Ωのとき，　γ(t,ω)＝γ(ω,t)＝t，

iv) γ(a^Xi*t2*…*tn,a^X2*u2*…*um)

＝a^γ(X1,X2)*γ(t2*…*tn,u2*…*um)，

v) 上記以外のとき，　γ(ti,tj)＝ρ(ti,tj)．

すると一般化項はγ(t,u)である．これをgen(t,u)で表わす．ρの決め方によって一般化項は複数できるが，それらは変数の改名で同値である．

〔定義〕項の順序

2つの整項tとuの間の順序を定義する：

t≦u ⇔ uとgen(t,u)は変数の改名で同値である．

このときuはtより一般的であるという．

4.3 テーブルとデータベース

この節では整項からテーブルとデータベースを定義する．

〔定義〕テーブル項

整項から構成される標準形t＝t1+…+tn の任意のtiとtjどうしが両立するとき，t をテーブル項という．テーブル項に含まれる変数は整項どうしで互いに共有されていないことを，本稿では仮定しておく．またt_set(t)の極大元の集合{i1',…,in'} からなるテーブル項t1'+…+tn'をt↑で表わし，それを標準テーブル項という．これは一意に決定することができる．以降テーブル項とはこの標準テーブル項を指しているものとする．そして整項 s＝a1^X1*…*an^Xnが，任意の1≦i,j≦nに対し

i) Xi∈Vで，Xiはtに含まれないこと，

ii) i≠jならばXi≠Xj，および

iii) 任意のtiに対し，ti≦s

となるときtのスキーマ項であるという．sとtの対(s,t)をテーブル，そしてテーブルの集合をデータベースと呼ぶ．

〔定義〕テーブルの射影

整項t によるテーブルT＝(s,t1+…+tn)の射影proj(t,T)は，tがsの部分項である場合に，(t,uni(t,t1+…+tn)) と定義する．ただしテーブル項t＝t1+…+tmとu＝u1+…+un の単一化は次のように定義されているとする：

uni(t,u)＝(uni(t1,u1)+…+uni(t1,un)+

…+uni(tm,u1)+…+uni(tm,un))↑．

テーブル項の定義は容易に整木集合に反映させることがで

きる．テーブル項の割当てを

vii) t＝t1+…+tnのとき，ι{t}＝ι{t1}∪…∪ι{tn}]

とする．

〔定理〕項の演算(1)-(9)は，上の拡張された意味関数と整木上の演算によって，整木の集合上でも成り立つ．

項のテーブルは整木の集合としてのテーブルに定義に対応できる．そこで項のテーブル(s,t1+…+tn)と整木のテーブルの関係を考えよう．すべての整項tiをスキーマ項の両立化項ti' に変換すると，ti'＝sθiなる代入θiが存在する．つまりスキーマ項の具体例がti' であり，その中の未定義定数の部分を省略したのがtiである．未定義定数ωは，整木では関数の未定義となっている．さらに値としてのφ(⊥)は値がないことを表わしている．またレコード(整項)の順序での極大元とは，より多くの情報をもったレコードのみを残すことを意味している．そしてテーブルの中のレコードはどの2つを取っても順序づけができないようになっている．

4.4 データベース操作

この節では通常のデータベース操作に対応する演算をテーブルの演算として定義する．互いに両立し変数を共有していないテーブルをT＝(s1,T1)とT'＝(s2,T2)，そしてs1＝s11*…*s1n，s2＝s21*…*s2m，T1＝t11+…+t1n，T2＝t21+…+t2mとする：

- 加算：s1＝s2のとき，加算T+T'＝(s,T1+T2)は，次のように定義される：
  T1+T2＝(t11+…+t1n+t21+…+t2m)↑．
- 減算：s1＝s2のときT−T'＝(s,T1−T2)は，次のように定義される：
  T1−T2＝t_set((T1+T2)↑)−t_set(T2)で構成される項．
- 乗算(multiplication)：s1とs2が共通の接合属性をもたない（そうでないときには属性名は適当に改名されているとする)とき，T×T'＝(s1×s2,T1×T2)は
  s1×s2＝s1*s2＝s11*…*s1n*21*…*2m，
  T1×T2＝t11*t21+…+t11*t2m+…+t1n*t21+…+t1n*t2m
  と定義される．ただしti*tjはti＝u1*…*up，tj＝v1*…*vqのとき，u1*…*up*v1*…*vqである．
- 除算(division)：　s2がs1の部分項であるときに，除算T÷T'は，sを，s1＝s2*sでs2と共通の接合属性をもたないs1の部分項とすると，T÷T'＝(s,T1÷T2)は
  T1÷T2
  ＝(s,(proj(s,T1)−proj(s,(proj(s,T1)×T2−T1)))↑)
  で定義される．

通常の合併，差，デカルト積そして除算はそれぞれ，上の+,−,×,÷に対応している．そして共通部分は，s1＝s2のとき定義される：

T∩T'＝(s,(T1+T2)−((T1−T2)+(T2−T1)))．

次にθ-選択とθ-結合を定義しよう．その前に制約と条件式を定義しておかなければならない：

〔定義〕制約と条件式

θを算術演算子(＝,＜,＞,≦または≧)あるいは集合演算子(＝,⊂,⊃,⊆または⊇)，そしてx,y∈V，c∈Δとするとき，xθcおよびxθyを制約という．また制約の変数を含む整項と制約の組(a'x,xθc)あるいは((a'x)*(a''y),xθy)を条件式という．x,yがΔの要素に具象化されたとき，条件式は評価される．本稿ではその内容には立ち入らないで，制約Cの評価をeval(C)と書く．eval(C)の結果はtrueまたはfalseである．

次に単一化を制約付の単一化に拡張する：

〔定義〕制約付単一化

単項tとuの制約(C)付単一化とは，C中の変数wguによってが具象化されたとき，eval(C)の結果で単一化の成否を決定することである．制約付単一化をc_uni(t,u|C)で表わす．これは項の遅延評価が必要なことを示している．

- θ-選択：　Tの条件式C＝(t,C')による選択sel(C,T)は，tがsの部分項であるとき定義され，tとの共通の接合属性が同じ変数をもつs1の改名項をs'（T1の変数とは素とする)とすると，次のように定義される：
  sel(C,T)＝(s,c_uni(s',T1|C')↑)．
- θ-結合：　乗算と同じ条件のTとT'の条件式C＝(u,C')＝(a'x*a''y,xθy)によるθ-結合は，uのa,a'がそれぞれs1,s2の接合属性であるとき，次のように定義される：
  join(C,T,T')＝(s1×s2, sel(C,(s1×s2,T1×T2))．

次にネストとアンネストをテーブル項の上で定義する．まず行ネスト／行アンネストのために

(3)　a0'(a1't1+…+an'tn)＝a0'(a1't1)+…+a0'(an'tn)

を適用する．さらにti∈Δのとき次の規則を導入する：

(10) e't1+…+e'tn＝e'(t1∪…∪e'tn)．

'＝'を左から右への書換えと考えれば，これは行ネスト操作に対応し，逆に右から左への書換えと考えれば，行アンネスト操作となっている．

(10)を追加することによって4.1節の単項標準形が一意的に求められない，つまり行アンネストは合流性をもつが，行ネストは非合流である．しかし対応する整木集合はすべて同一である．行ネストの結果の一意性を保証するためにいくつかの提案が従属性制約に基づく順序の導入という形でなされている[Arisawa83, Kambayashi 83]が，本稿ではそれには立ち入らない．

行ネスト操作の結果に対する単一化と一般化を考える:

〔定義〕単一化と一般化の拡張

両立する2つの整項はレコード値をもった場合でもt＝a1*t1*…*an*tn, u＝a1*u1*…*an*unと仮定することができる(レコード値としての項も両立する)ので，それらの単一化と一般化をそれぞれ

uni(t,u)=a1*uni(t1,u1)*…*an*uni(tn,un),

gen(t,u)=a1*gen(t1,u1)*…*an*gen(tn,un)

と定義する．すると単一化と一般化は，行ネスト／行アンネストの結果に関わらず，等しい．

次に列ネスト／列アンネストを定義する:

・列ネスト: スキーマ項sとそれに含まれるいくつか属性a1,…,an∈attr(s) に対し，各aiをa*aiで置換える操作ζ＝(a1,…,an)→a*(a1,…,an)をスキーマ項に対する列ネスト という．ただし¬a∈attr(s)の場合にのみ定義される．テーブルTに対する列ネストはT[ζ]と書く．

・列アンネスト: 列ネストとは逆にa*a1,…,a*anをそれぞれa1,…,anで置換える操作ζ'＝a*(a1,…,an)→(a1,…,an)を列アンネストといい，T[ζ']と書く．

4.4 レコード構造による演繹

この節ではtで定義した項を使った制約付論理型言語を定義し，それを使用したデータベース操作を述べる．データベース操作を定義するためには否定の導入が必要である．

〔定義〕リテラル

整項tあるいはそれに否定を付けたもの¬tをリテラルという．tを正のリテラル，¬tを負のリテラルという．

〔定義〕プログラム

整項t と制約の集合C＝{C1,…,Cm}とリテラルの有限集合B＝{t1,…,tn}の3つ組をプログラム節といい，t←C1,…,Cm|t1,…,tnと記す．tをヘッド，Cを制約部，Bをボディと呼ぶ．とくにボディと制約部が空集合のとき単位節といい単にt と書く．プログラム節の集合がプログラムである．

そしてプログラムのモデルと変換は通常通り定義される．ところが上の一般的なプログラムの最小モデルは一意的ではないので，否定と再帰の使用を限定した，データベース操作を展開するのに充分なサブクラスを定義する:

〔定義〕層化(stratified)プログラム〔Apt 86〕

プログラムPが次のように分割できるとき，Pを層化プログラムという: P=P1∪…∪Pn (P1,…,Pnは互いに素)で，

i) Piの節の正のリテラルは$\bigcup_{j\leq i}$ Pjで定義されている，

ii) Piの節の負のリテラルは$\bigcup_{j<i}$ Pjで定義されている．

本稿のプログラムはすべて層化プログラムに限定する．

〔定義〕層化プログラムのモデル

まずプログラムPに対してTP↑ωを次のように定義する:

TP↑0(M) ＝M,

TP↑n+1(M)＝TP(TP↑n(M))∪TP↑n(M),

TP↑ω ＝$\bigcup_{n=0}^{\infty}$ T↑n(M).

そして層化プログラムP=P1∪…∪Pnに対してモデルMPを定義する: M1＝TP1↑(φ),…,Mi＝TPi↑(Mi-1),…,

Mn＝TPn↑(Mn-1)=MP.

するとMPは最小モデルとなっている．

次にプログラムの解について述べる:

〔定義〕論理的帰結

任意の変数割当てに対し，項Gが，プログラムPの任意のモデルに含まれているならば，つまりMPに含まれているならば，GはPの論理的帰結であるという．

そしてプログラムの(制約付)ゴールと解代入については通常通り定義し，次にプログラムの導出を定義する:

〔定義〕SLDNF導出列

プログラムP をΓ(H1,E1,B1),…,(Hn,En,Bn)}とし，ゴールを(C,G)とする．このときD0 ＝(C,G,φ),

Di+1＝(Ci+1,Gi+1,θi+1),

ただし，Gi＝{Gi1,…,Gim}とすると，あるGijがあってGijが正のリテラルならば，

θi+1＝θiζ，Gijζ＝Hkζ，

Gi+1＝(Gi-{Gij})∪Bk，Ci+1＝sat(Ciζ∪Ekζ)，

Gijが負のリテラル(¬P)ならば，

Pは具象化項でどのHiとも単一化できず，

Ci+1＝Ci，Gi+1＝Gi-{Gij}，θi+1＝θi

なるD0,…,Dn,…の列をSLDNF導出列という．導出列のあるDnで，Cn＝φかつGn＝φならば，この導出列は成功といい，θnをゴール(C,G)の解という．またCnに充足不可能な制約が含まれているならば，この導出列は有限で失敗という．導出列は，成功か，有限で失敗か，あるいは無限列となる．

このとき次の定理が成り立つ:

〔定理〕整合性

プログラムPに対して成功のSLDNF導出列(C,G,φ)⊢…⊢(φ,φ,θ)が存在するとき，任意のリテラルg∈G は，gが正のとき，任意の変数割当てに対しt(gθ)∈MPである．gが負(¬t)のとき，任意の変数割当てに対して¬t(tθ)∈MPである．すなわち解θは解代入となっている．

これでデータベース操作を定義する準備ができたので，4.4 節の操作に対応して以下データベース操作の定義をおこなう．テーブルをT1＝(s1,t1), T2＝(s2,t2)とする．各テーブルのスキーマ項とテーブル項（テーブル項は単位節

に分解されている)にはテーブルを識別するための属性a1，a2がそれぞれ付加されている．すると関係モデルとPrologの関係同様に，実行結果は整項tに1つずつ返される：

・射影proj(t,T1)：tをs1の部分項であるとすると
　　t←a1^s1.
・合併T1∪T2：t＝s1＝s2とすると
　　t←a1^s1.
　　t←a2^s2.
・差T1－T2：合併と同じくt＝s1＝s2とすると
　　t←a1^s1,¬a2^s2.
・デカルト積T1×T2：t＝s1*s2とすると
　　t←a1^s1,a2^s2.
・除算T1÷T2：tとs2は素でs1＝t*s2 とすると
　　t ←a1^s1,¬b^t.
　　b^t←a1^s1,a2^s2,¬a1^s1.
・共通部分T1∩T2：t＝s1＝s2 とすると
　　t←a1^s1,a2^s2.
・θ-選択sel((t,C),(s1,T1))：t＝s1とすると
　　t←C|a1^s1.
・θ-結合join((t,C),T1,T2)：t＝s1*s2とすると
　　t←C|a1^s1,a2^s2.
・列ネストT1{ζ}：t=s{ζ}とすると
　　t←a1^s1.
・列アンネストT1{ζ'}：t=s{ζ'}とすると
　　t←a1^a1.

しかし行ネスト／行アンネストはこのままでは処理できないし，またtに返される整項全体には重複の可能性がある．そこでプログラムの拡張を，整項からテーブル項に，そしてレコード値をもつ形に拡張することが考えられる．このようなテーブル項に基づいたプログラムが可能な条件は現在まだ検討中である．

5．おわりに

非正規形モデルは，多くの応用でひじょうに優れた性質をもっている．ICOTで進めているKappa は，さまざまな知識情報処理システムのために，非正規型モデルを内部モデルにして研究開発されている．本稿の枠組は，それに対応した理論面からの検討である．

しかしデータベースの証明論的扱いのためには，閉領域(domain closure)公理や外延(extension) 公理など，現実的には実装不可能な公理を多く必要とする．そこで演繹データベースを考える場合，理論面と工学面を明確に区別しなければならない．現在その点を含め，非正規形への演繹的アプローチとしてはさらに検討を続けている：

i）レコード値への拡張，
ii）テーブルを基にしたプログラムへの拡張，
iii）タクソノミーのような構造の取扱いと継承の処理，
iv）質問の最適化，あるいは工学的検討

などである．

現在Kappa の試作システムの実装をほぼ終了し，次期システムの基本検討を開始している．試作システムでは，通常の非正規形モデルの機能のほかに，項の扱いやタクソノミーの構造と操作をサポートしている．そして次期システムでは本稿で述べた枠組を中心にして，演繹機能の付加と高度の知識表現を目指した知識ベースを展開しようとしている．今後さらに検討を続け，そして現実のKappa システムに反映させる予定である．

謝辞

本研究は，知識ベース管理システムKappa の研究開発の一環として，内田第４研究室長の下でおこなわれている．日頃，暖かい助言を頂いている横井次長，内田室長，貴重な助言を頂いている向井主任研究員，同僚の金枝主，河村各氏のほか，Kappa，CKLのメンバーの各氏に感謝する．

参考文献

[Abiteboul 84] Abiteboul,S. and Bidoit,N., "NonFirst Normal Form Relations: An Algebra Allowing Data Restructuring", INRIA TR-347, 1986.
[Ait-Kaci 84] Ait-Kaci,H., "A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures", Dissertation, Univ. of Pennsylvania, 1984.
[Ait-Kaci 86] Ait_Kaci,H. and Nasr,R., "LOGIN: a Logic Programming Language with Built-in Inheritance", J. Logic Programming, vol.3, pp185-215, 1986.
[Apt 86] Apt,K.R., Blair,H. and Walker,A., "Toward a Theory of Declarative Knowledge", Proc. of the Workshop on Foundation of Deductive Database and Logic Programming, 1986.
[Arisawa 83] Arisawa,H., Moriya,K. and Miura,T., "Operations and the Properties on Non-First-Normal-Form Relational Databases", 9th VLDB, pp197-204, Florence, 1983.
[Fischer 83] Fischer,P.C. and Thomas,S., "Operations for Non-First-Normal Form Relations", COMPSAC 83, pp464-475, 1983.
[Gallaire 78] Gallaire,H. and Minker, J.(ed), "Logic and Data Bases", Plenum, 1978.
[Gallaire 84] Gallaire,H., Minker,J. and Nicolas,J.-M., "Logic and Databases: a Deductive Approach", Computing Surveys, vol.16, no.2, pp153-185, 1984.
[Jacobs 82] Jacobs,B.E., "On Database Logic", J. ACM, vol.29, no.2, pp310-332, 1982.
[Kambayashi 83] Kambayashi,Y., Tanaka,K. and Takeda,K., "Synthesis of Unnormalized Relations Incorporating More Meaning", Int.J.Information Science, vol.29, pp201-247, 1983.
[Maier 86] Maier,D, "Logic for Objects", Proc. of the Workshop on Foundation of Deductive Database and Logic Programming, 1986.
[Makinouchi 77] Makinouchi, A., "A Consideration on Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model", 3rd VLDB, pp447-453, Tokyo, 1977.
[Mukai 86] 向井，"自然言語処理のための単一化拡張と遅延的実行制御(ドラフト)"，ICOT内部TR,1986.
[Pistor 86] P.Pistor and R.Traunmueller, "A Database Language for Sets, Lists and Tables", Inform. Systems, vol.11, no.4, pp323-336, 1986.
[Reiter 84] Reiter,R., "Toward a Logical Reconstruction of Relational Database Theory", in Brodie,M.L., Mylopoulos,J and Schmidt,J.W.(ed.), "Conceptual Modelling", pp191-238, Springer, 1984.
[Schek 82] Schek,H.-J. and Pistor,P., "Data Structures for an Integrated Data Base Management and Information Retrieval System", 8th VLDB, pp197-207, Mexico City, 1982.
[Yokota 86] 横田，内田，溝口，"知識ベース管理システムKappaの概想―PSI の環境と設計方針"，情報処理学会第33回全国大会，5M-5，1986.
[Yokota 87a] Yokota,K., Kanaegami,A., Kawamura,M. and Uchida, S., "KAPPA― Knowledge Base Management System on PSI", ICOT TM, to appear, 1987.
[Yokota 87b] 横田，"非正規形モデルの論理に向けて"，ICOT内部TR，1987.