

論理関数の最小化の手法を利用したリレーショナルデータベースの正規表分解について

京都大学 工学部

上林 彌彦

1. まえがき

本論文はリレーショナルデータベースの正規表分解の問題を、関係数の最小化という条件のもとで考察し、従来の方法の問題点を指摘するとともに、あらたに論理関数の素項生成の手法を用いた方法を提案している。

Codd^[1]によって提唱されたデータベースのリレーショナルモデルは、利用者のデータモデルと計算機の記憶構造との間の独立性が高く、多様なデータの扱いが可能であることから有効なデータベースの実現法であると考えられている。この方法ではデータ間の関係はすべて表の形で表現されるためデータの追加・削除・更新が矛盾なく行なえるような表集合を求めることが問題となる。このため各データの属性の間にある関数従属関係に注目して、正規と呼ばれる条件(いくつか知られているが第3正規形がよく用いられる)を満足するような表集合を求める方法について研究がなされている^{[1][2]}。

Delobel, Casey^[3] は, 関数従属関係と論理関数との間の対応関係を利用してこの問題を扱った。また, Wang, Wedekind^[4] は, 正規形の関係集合を求めるためにもとの関数従属関係集合から冗長なものを取り除いた非冗長な関数従属関係の集合を用いる方法を示した。最近, Bernstein^[5] はこれらの方法を発展させるとともに, [3][4]の論文の問題点の指摘も行っている。文献[5]の中の重要な点は, 非冗長な関数従属関係と第3正規形との対応関係を与えた Delobel, Casey らの結果に対する反例の存在を示した点と, 等価な key の存在する場合の扱いを導入した点である。しかし, Delobel, Casey らの結果に対する反例が存在することから第3正規形であるためのある十分条件を満足する関係のみを扱った点に問題がある。また, 与えられた関数従属関係集合から冗長なものを順次取り去るのみで関係数の最小化を行なえるという点が論文の内容的な特色となっているが, 実際には等価な key を求めたり, 第3正規形であるための十分条件の判定のためにもとからは与えられていないが導き出すことのできる関数従属関係を使う必要がある。本論文では, 論理関数の素項生成の方法を用いてすべての素な関数従属関係を key が同じものを同時に生成するという形式で求め, これらを用いて関係数の最小化を行なう方法について述べている。

この方法によれば、等価な key の判定や Delabel, Casey の条件を満足しない場合の判定も容易であり、第3正規形だけでなく、Boyce-Codd の正規形で最小のものを求めるような問題にも適用できる。

以下の議論では、文献[3][4][5]と同様、各関数従属関係は1つの意味しか持たないとして仮定する。Codd の第1正規形で与えられた1つの表を第3正規形に分解する等のときにはこの仮定が成立する。また、Fagin によって導入された multi-valued dependency の問題についても文献[3][4][5]と同様に考えに入れたいこととする。

2. 基本的事項

本節では、リレーショナルデータベースとその最小関係表現のために必要な事項をまとめる。

関係 R_j は、各列が属性、各行が異なるデータの組に対応する表の形で与えられる。各属性を A_i, B_i, C_i, D_i 等で表わし、属性集合を W_i, X_i, Y_i, Z_i 等で表わすものとする。 R_j が属性 A_1, A_2, \dots, A_n の間の関係を表わすとき、これを $R_j(A_1, A_2, \dots, A_n)$ で表わす。関係 R_j の内容は、データの追加・削除・更新による時間的に変化してゆくものであるが、ふつうある種の制限条件を満足して

いることが多い。属性 A_1, A_2, \dots, A_k の値が組合せを 1 つ決めると属性 B_1 の値も必ず決まるといふ制限条件が常に成立している場合、 B_1 は A_1, A_2, \dots, A_k に対して関数従属であるといふ。関数従属関係は $f: \{A_1, A_2, \dots, A_k\} \rightarrow B_1$ のような形で表現される。 $\{A_1, A_2, \dots, A_k\} \rightarrow B_i (1 \leq i \leq k)$ であるとき $f: \{A_1, A_2, \dots, A_k\} \rightarrow \{B_1, B_2, \dots, B_k\}$ のような表現を用いる。集合の要素数が 1 のときは $\{ \}$ は省略される。また、関数従属関係の正しいことを明記したい場合は、 $\{A_1, A_2, \dots, A_k\} \rightarrow B_1$ のような表現を用いる^[2]。

たとえば、関係 $EMPDEPT(NAME, DEPT, MGR)$ は、従業員名、部門名、部長名からなる表で表現される。各従業員は 1 つの部門に属し、各部門には 1 人の部長がいる（部長はいくつかの部を兼任できる）場合、全関数従属関係として、 $NAME \rightarrow DEPT, DEPT \rightarrow MGR, NAME \rightarrow MGR$ がある。この場合、 $NAME \rightarrow MGR$ という関数従属関係は他の 2 つの関係から導くことができるので、基本的な関数従属関係としては $NAME \rightarrow DEPT$ と、 $DEPT \rightarrow MGR$ だけを考えることができる。逆に、関数従属関係の集合が与えられた場合に、それらを用いて種々の関数従属関係を作り出すこともできる。

Delobel, Casey^[3] は、関数従属関係間の 6 つの変換を示

したが、さらに Armstrong^[6]は、冗長なものを除いて公理化を行っている。たとえば、与えられた関数従属集合から導かれるすべての関数従属集合は、次の(1)(2)(3)の適用のみによ、て導かれることが示されている。

$$(1) X \rightarrow Y$$

$$(2) X \rightarrow Y \text{ ならば } X \cup V \rightarrow Y$$

$$(3) X \rightarrow Y \text{ で } Y \cup Z \rightarrow W \text{ ならば } X \cup Z \rightarrow W$$

この(3)は疑似推移律と呼ばれ、これはすでに述べた $X \rightarrow Y$, $Y \rightarrow Z$ なら $X \rightarrow Z$ という推移律を含むもので、たとえば、 $\{A_1, A_2\} \rightarrow B$, $\{A_2, B\} \rightarrow C$ なら $\{A_1, A_2\} \rightarrow C$ が導かれるものである。この疑似推移律が関数従属関係の扱いを複雑にする原因となっている。

Delobel, Casey^[3]は、関数従属関係を扱うために、論理関数との対応に注目した。すなわち、各属性 A_i, B_i, C_i, D_i 等に対応する変数 a_i, b_i, c_i, d_i を考え、 $\{A_1, A_2, \dots, A_k\} \rightarrow B_1$ が全関数従属関係ならば、これに対応して論理積 $a_1, a_2, \dots, a_k \rightarrow b_1$ を作る。与えられた関数従属関係に対応するすべての論理積の論理和をとり、これを関数従属関係集合 F に対応する論理関数 \hat{F} とする。論理関数 \hat{F} へ、 $c_1 = c_2 = \dots = c_n = 1$, $d_1 = 0$ を代入すると $\hat{F} = 1$ とすれば、関数従属関係 $\{C_1, C_2, \dots, C_n\} \rightarrow D_1$ が導かれることを示している(と

の C_i を 0 にしても成立しなければ全関数従属関係となる)。これは、論理積の間の変換と上記の関数従属関係間の変換が 1 対 1 に対応していることを利用したものである。たとえば Armstrong の公理の (2) の例として、 $A \rightarrow B$ ならば $\{A, C\} \rightarrow B$ が成立することは、 $a\bar{b} = a\bar{b} + ac\bar{b}$ といふ事実に対応している。(3) の例として、 $A \rightarrow B$, $\{B, C\} \rightarrow D$ ならば $\{A, C\} \rightarrow D$ が成立することは、 $a\bar{b} + bcd = a\bar{b} + bcd + abc\bar{d} + abc\bar{d} = a\bar{b} + bcd + ac\bar{d}$ といふ事実に対応している。

与えられた関数従属関係の集合を F とし、 F から導かれるすべての関数従属関係を含まれる集合を F の閉包と名付け F^+ で表す。逆に上記の規則による冗長な関数従属関係を含まない集合を非冗長であるという。関数従属関係の集合 G が非冗長であって $G^+ = F^+$ である場合 G を F の非冗長カバーという。非冗長カバーの中には、 $G \subseteq F$ が成立しないものもありうる。とくに非冗長カバーのうち要素数が最小のものを最小カバーと呼ぶ。 F の最小カバーは、論理関数化の最小化を行った結果をそのまま利用できる。

各ドメインが関係ではなく単なる値に対応するような関係の表現を第 1 正規形という。任意の関係は簡単に第 1 正規形にできる。

関係 $R(A_1, A_2, \dots, A_n)$ において, $X \subseteq \{A_1, \dots, A_n\}$ が R の *key* であると言われるのは, $X \rightarrow A_i$ がすべての $A_i \notin X$ に対して成立し, X のどの部分集合もこのような性質を持たない場合である。Key は一意的ではないので, 可能などれかの *key* に含まれる属性を特に素であるという。素でない属性は非素であるといわれる。EMPDEPT(NAME, DEPT, MGR) の例では, NAME がこの関係の *key* となる。以下では *key* となる属性を区別する必要がある場合, 属性名に下線を付すものとする。

関係表現が第2正規形であるというのは, それが第1正規形であって, かつ, すべての非素な属性が, 任意の *key* に対して全関数従属である場合にいう。

上記の EMPDEPT は第2正規形の例となっている。しかしながら, ある部門に属する従業員がすべてやめてしまうと, その部門と部長に関する情報も失われてしまう。すなわち, 部下の1人もいない部長に関する情報が消えてしまう。第3正規形は, このような問題を解決するために Codd によって提案されたものである⁽²⁾。

関係表現が第3正規形であるというのは, それが第2正規形であって, どの非素属性も, すべての *key* に対して推移従属をしていない場合にいう。ここで A_i が X に推移従属する

とは $R(A_1, \dots, A_n)$ において $X \rightarrow Y$, $Y \rightarrow X$ で $Y \rightarrow A_i$ となるような Y が存在する場合である ($A_i \notin X$, $A_i \notin Y$)。

任意の関数従属関係から第3正規形の関係表現集合を求めることができることが Codd によって示されているため、与えられた関数従属集合から第3正規形であるような関係数が最小の関数表現の集合を求めるのは1つの興味ある問題である。この問題に関しては、文献[3][4][5]等で扱われているが、関係数最小という点では不十分な点がある。

Delobel, Casey^[3]によれば、与えられた関数従属関係集合に対して、対応する論理関数 f を求め、 f を素項の和の形で表わせば(最小カバー)、関係数の最小化が得られることになる。この最小カバーの関数従属関係は第3正規形の条件である推移従属性を持たないことが示されていたが、

Bernstein はその反例を作っている。

Wang, Wedekind^[4]は、与えられた関数従属関係集合から冗長な関数従属関係を取り除いて非冗長な関数従属関係集合を求め、これらのうち左辺の等しいものをまとめて関係集合を得るとい手法を提唱している。

Bernstein^[5]のアルゴリズムも本質的には同じであるが、文献[4]の手法で問題となつた等価な key の扱いや、文献[3]の反例に対応する部分に関して詳細な議論を行っている。

Codd の第3正規形では、非素な属性には推移従属性を許していないが、素な属性に対して許しているため推移従属性の持つ諸問題は残されている。Boyce-Codd の定義による第3正規形(Boyce-Codd 正規形という)は、素な属性に対しても、そのような制限を付加したもので、つぎのように述べられる。

関係 R の中のある属性集合 X に対し、 X に含まれないある属性 A に対し、 $X \rightarrow A$ という関数従属関係があれば、 X に含まれない任意の属性 B に対しても $X \rightarrow B$ という関数従属関係がある。

この定義を満足するような正規形表現が望ましいが、実現には、 key を構成する属性の部分集合が非素な属性に従属している、いわゆる key 破壊のおこる場合は非常に困難であることが知られている。

3. 関係数最小化における問題点

本節では、文献[3][4][5]の問題点について考察する。

まず、文献[3]では、論理関数の最小化の手法を用いて関数従属関係の最小化を行っている。しかし、関係数最小化のためには最小カバードけではなく最小でない非冗長カバード考える必要があり、論理関数の非冗長カバードを求めなければなら

らない。

文献[4]では、与えられた関数従属関係集合 F から冗長なものすべてを可能な組合せで除き、可能な非冗長カバーの集合を求めようとしているが、与えられた関数従属関係集合によっては関係数の多いものが求まる可能性がある。したがって、最小のものを求めるには、 F^+ を考えなくてはならない。

{例 1} 与えられた関数従属関係がつぎに示すものであるとする。

$$\{A, G\} \rightarrow D, \quad \{B, G\} \rightarrow E, \quad \{C, G\} \rightarrow F, \quad A \rightarrow B \\ B \rightarrow C, \quad \{C, G\} \rightarrow A$$

これは非冗長でどの関数従属関係も取り除くことはできない。対応する関係は、*key*の同じ部分をまとめることにより次のようになる。

$$R_1(\underline{A}, B), \quad R_2(\underline{B}, C) \quad R_3(\underline{A, G}, D) \\ R_4(\underline{B, G}, E), \quad R_5(\underline{C, G}, A, F)$$

しかしながら、つぎのような関数従属関係も上記と同じ閉包を持つ。

$$A \rightarrow B, \quad B \rightarrow C, \quad \{C, G\} \rightarrow A \\ \{C, G\} \rightarrow D, \quad \{C, G\} \rightarrow E, \quad \{C, G\} \rightarrow F$$

この場合、関数従属関係数は同じだが関係数は減らされる。

$$R_1(\underline{A}, B), \quad R_2(\underline{B}, C), \quad R_3'(\underline{C, G}, A, D, E, F)$$

文献[5]では、 key の等価なものを先に求めて、 key の等価性に関係した関数従属関係はそのまゝにして、その他の部分で冗長なものを取り去るという方法を採用している。このような方法で関係数の最小化に対する保障が得られるが、つぎのような問題点もある。

(1)文献[5]の方法は、与えられた関数従属関係集合 F から冗長なものを取り除いて最小数の関係集合が求まり F^+ を用いない点が特色であるとしているが、 key の等価性の判定の際には F^+ の要素を考える必要がある。第3正規形の十分条件の判定にも F^+ の要素を考える必要がある。

(2)等価な key の部分に対する最小化は考えられていない。また、等価な key をまとめてしまうと第3正規形にたどり着く結局最小数の関係集合の得られない可能性がある。たとえば、 AB と CD が等価で全関数従属関係として、 $AB \rightarrow E$, $C \rightarrow E$ のある場合等が考えられる。

(3)等価な key を考えることにより、1つの関係に2つ以上の key の存在を許しているが、 key の要素に対しては第3正規形の推移従属性の制限がなくなる性質は使われていない。

以上のことから、本論文では、まず F^+ の中的全関数従属関係集合をすべて求め、それらの操作から第3正規形やBoyce-Coddの正規形を求める方法について検討している。 F^+ の中

のすべての全関数従属関係の集合が求められると次のような利点がある。

(1) 可能なすべての等価な key 集合を比較的容易に求めることができる。文献[5]には等価な key を求めるアルゴリズムは示されていない。

(2) $X \rightarrow Y$, $Y \rightarrow X$, $Y \rightarrow Z$, $X \rightarrow Z$ という形の推移従属のおこるすべての可能な場合を調べることができる。とくに「不可欠な関数従属性」を「取り去ると F^+ が変化するような関数従属関係」と定義しておけば、 $X \rightarrow Z$ が不可欠かどうかによって第3正規形になるかどうかを文献[5]のような十分条件でなく必要十分条件で調べることができる。

(3) ある関係を生成するとその中に含まれる属性間のみに関係した関数従属関係のうち $X \rightarrow A$ (A は素な属性) の形のものをすべて含ませることができる。不可欠な関係を用いて別の関数従属関係を減らすこともできる。たとえば、 $\{A, B\} \rightarrow C$ が不可欠なら、 $C \rightarrow B$ については考えなくてよい。

(4) さらに Boyce-Codd の正規形の生成のアルゴリズムも作りうる。

F^+ の中のすべての全関数従属関係を生成するのに論理関数の素項生成の方法を変形して F に適用し、同じ key になる関数従属関係は同じ部分 $tree$ 上に現われるようにして関係の生

成を簡単化した。

4. 木探索による関数従属関係の生成

木探索における問題解決の手法においては、与えられた問題をいくつかの部分問題に分けるというステップを繰り返して多数のより解き易い問題へと帰着させている。この方法はある問題に対応する節点の下にいくつかのその部分問題に対応した節点を並べて枝で結ぶという操作を繰り返して木の形に表わすことができる。この方法で重要なのは、部分問題に分ける方法(節点の枝出しの方法)と部分問題数を減少させるための工夫である。

論理関数 F において、いくつかの文字 (x_i および \bar{x}_i , $i=1, \dots, n$) よりなる論理積を P としたとき、 $P=1$ なら $f=1$ となるなら P は F の *implicant* であるという。Implicant $P' (\neq P)$ があって、 $P'=1$ なら $P=1$ となるとき P は非素であるという。非素でない *implicant* を素項という。

論理関数の素項を求める場合、つぎのようにして部分問題に分けていくことができる。すなわち、論理関数に表われる文字 L_i (x_i または \bar{x}_i) に 1 を代入して得られる関数を $f_{L_i=1}$ と表わす。 f の素項のうち L_i を含むものはすべて $f_{L_i=1}$ の素項と L_i との積に含まれてしまう。したがって

\mathcal{F} に含まれる各文字 L_i について $\mathcal{F}_{L_i=1}$ の素項を求めるという操作を繰り返してやればよい。

この方法では、素項はすべて導出できるが、非素項もかなり含まれること、同じ素項が何度も表われること等により、木が大きくなり効率もよくない。

Slagleらは、枝出しする文字に順序付けを行うことにより、この方法の効率を向上させた^[8]。著者らは、さらに和積形の1つの $clause$ に注目しながら枝出しを行う方法で、さらに効率の向上を行った^[9]。本論文のアルゴリズムは文献^[9]の方法を基礎としている。

関数従属関係の集合 F が与えられたとき \hat{F} の素項については、つぎの性質がある。

[補題1] 関数従属関係集合 F に対応する論理関数 \hat{F} の素項は F^+ に含まれるすべての全関数従属関係に対応している。

(証明) \hat{F} の素項集合と (F^+) の素項集合は同じであり、素項は全関数従属関係に対応することからあきらかである。(証明終)

[補題2] 関数従属関係集合 F に対応する論理関数 \hat{F} の事項はすべて否定文字を1文字しか含まない論理積となっている。

(証明) もとの論理積が否定文字を1文字しか含まないので素項を求めるコンセンサス法等を適用してもその性質が保存されるためである。(証明終)

補題2により, 節点に対応する論理関数が否定文字のみより成る場合以外は, 肯定文字のみに注目して枝出しを行えばよいことになる。

形式的なアルゴリズムを示す前に例題を示しておく。

〔例 2〕 つぎに示すような関数従属関係集合 F が与えられたとして, F^+ に含まれる全関数従属関係をすべて求めてみる。

$$\begin{aligned} \{A, D\} &\rightarrow C, & \{C, D\} &\rightarrow B, & \{B, C, D\} &\rightarrow A, \\ \{B, C\} &\rightarrow D \end{aligned}$$

対応する論理関数 F はつぎのようになる。

$$\hat{F} = ad\bar{c} + cd\bar{b} + bcd\bar{a} + bcd$$

\hat{F} を和積形式で表現すると

$$\hat{F} = (a+c)(c+d)(b+d)(\bar{a}+\bar{b}+\bar{c}+\bar{d})$$

この論理関数を木の根の節点に対応させる。枝出しは、和積形の要素となっていてあるどれか1つの和項 (clause という) に注目して行う。肯定文字の出現頻度を求めると

$$a \dots 1, \quad b \dots 1, \quad c \dots 2, \quad d \dots 2$$

となっているため, $(c+d)$ をその項に含まれる文字の頻度和が最大のものとして選択する。

$c = 1$ とすれば関数は,

$$\hat{F}_{c=1} = d\bar{b} + bd\bar{a} + b\bar{d} = (b+d)(\bar{a}+\bar{b}+\bar{d})$$

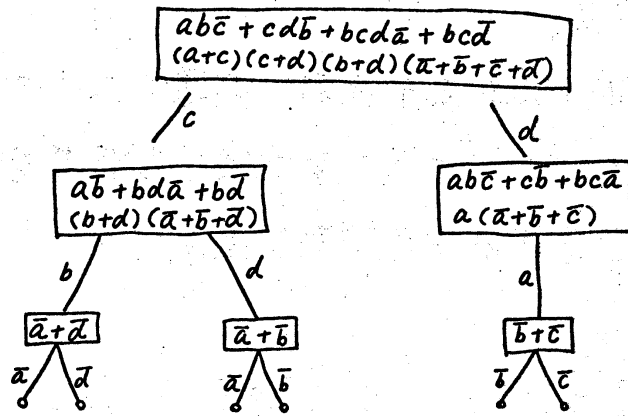


図1 F^n の全関数従属関係を求める方法

となるので、さらに b と d による枝出しを行う。

$$\hat{F}_{c=1, b=1} = \bar{a} + \bar{d}$$

$$\hat{F}_{c=1, d=1} = \bar{a} + \bar{b}$$

$\hat{F}_{c=1, b=1}$ は否定項だけからなるので否定項による枝出しを行う。関数値が 1 になると枝出しをやめる。図 1 にこの場合の木を示している。木の根から枝を順次たどって葉の方へ向いながら、枝のラベルを結合してゆけば、関数従属関係に対応した論理積が得られる。すなわち、

$$cb\bar{a}, cb\bar{d}, cd\bar{a}, cd\bar{b}, ad\bar{b}, ab\bar{c}。$$

F^+ の全関数従属集合は、

$$\{C, B\} \rightarrow \{A, D\}, \quad \{C, D\} \rightarrow \{A, B\},$$

$$\{A, D\} \rightarrow \{B, C\}$$

となる。

この方法の特色は、key の等しい全関数従属関係を同じ部分木で生成しているため、関係を作るのが容易な点である。

[アルゴリズム 1] 与えられた関数従属関係集合 F に対して、 F^+ に含まれる全関数従属関係をすべて求める方法

(1) 与えられた関数従属関係集合 F に対応する論理関数 F を和積形で表現する。

(2) 和積形の各和項に含まれる文字の集合を *clause* と名付け、すべての和項に対する *clause* 集合を木の根の節点

に対応させる。以下 a_i とその否定 \bar{a}_i は異なる文字として扱う。

- (3) 木の非終端節点のうち枝出しの行われていない節点を選びこれを S とする。 S の各 $clause$ について、それに含まれる肯定形の文字の $clause$ 集合内での頻度重をその $clause$ の重みとする。 S から否定文字数最小、文字数最小、重み最大という条件で1つの $clause$ を選ぶ。この中の肯定文字を $clause$ 集合中の頻度順に、否定文字をその後並べたものを $O_S = \{L_1, L_2, \dots, L_i, \dots\}$ とする。
- (4) O_S の順に従って各 L_i について、 L_i とラベル付けされた枝出しを行いその端点を S' とする。 S' に対応する $clause$ 集合は、つぎのようにして決められる。(a) S の L_i を含む $clause$ はすべて除く。(b) 残った $clause$ から L_1, \dots, L_{i-1} のうち肯定文字を除く。ここで、(a) で $clause$ 集合が空になれば S' を有効節点と呼び、(b) で空になる $clause$ ができると S' を無効節点と呼ぶ。それ以外の場合は非終端節点と呼ぶ。

(5) 上記の(3), (4)を非終端節点が無くなるまで繰り返す。

- (6) 生成された木の根より葉までたどる道に含まれる枝のラベルの論理積に対応する関数従属関係が F^+ に含まれる全関数従属関係をすべて含んでいる。

5. 関数従属関係の生成法を利用した第3正規形の構成

文献[5]では、等価な key を求めて関係数最小化に利用しているが、 F^+ における等価な key をすべてを求める方法は示されていない。 X と Y とが等価な key のとき、 $X \sim Y$ と表わすことにすると、つぎのような性質がある。

$$(1) X \sim X$$

$$(2) X_1 \sim Y_1, X_2 \sim Y_2 \text{ なら } X_1 \cup X_2 \sim Y_1 \cup Y_2$$

したがって、 $X \sim Y$ であって、 $X' \subset X, Y' \subset Y, (X, Y) \neq (X', Y')$ の場合に $X' \not\sim Y'$ であるような $X \sim Y$ を非常長な等価性と呼ぶ。与えられた関数従属関係集合 F から導かれるすべての非常長な key の等価性を求めることが重要である。この場合、一般に $AC \sim BC$ であって $A \not\sim B$ であることに注意しておく。

〔例 3〕 図 1 に示される例について、すべての非常長な key の等価性を求めてみる。同一 key となる各部分木について、

$$\{C, B\} \rightarrow \{A, D\}, \{C, D\} \rightarrow \{A, B\}, \{A, D\} \rightarrow \{B, C\}$$

これに冗長な関数従属関係 ($A \rightarrow A$ 等) を入れると、右辺も左辺の集合を加えることになり、

$$\{B, C\} \rightarrow \{A, B, C, D\}, \{C, D\} \rightarrow \{A, B, C, D\}$$

$$\{A, D\} \rightarrow \{A, B, C, D\}$$

この場合は、

$$\{B, C\} \rightarrow \{A, D\}, \quad \{A, D\} \rightarrow \{B, C\}$$

$$\{B, C\} \rightarrow \{C, D\}, \quad \{C, D\} \rightarrow \{B, C\}$$

となるため、

$$\{A, D\} \sim \{B, C\} \sim \{C, D\}$$

となる。このため、 $R(A, B, C, D)$ の各属性は素となるためこのままで第3正規形となっている。

この方法をグラフを用いて形式的に表わすと、つぎのようになる。

〔アルゴリズム2〕 すべての非冗長な *key* の等価性を求める方法

- (1) F に対応する木をアルゴリズム1で求め、同一 *key* となる各部分木について関数従属関係 $X \rightarrow Y$ を求める。
- (2) $X \rightarrow Y$ を冗長な関数従属関係も含めて、 $X \rightarrow X \cup Y$ とする。
- (3) $X \rightarrow Y$ と $X' \rightarrow Y'$ があり、 $X' \subset X$ ($X' \neq X$) なら $X \rightarrow Y$ を $X \rightarrow Y \cup Y'$ で置き換える。
- (4) 各々の関数従属関係に対応して左辺をラベルとする節点を作る。節点 α に対応する関数従属関係の右辺に含まれる集合をラベルとする節点 β があれば、 α より β へ有効枝をひく。
- (5) 相互に到達可能な節点に対応するラベル同士は等価である。

アルゴリズム 1 の端節点の関数を解析すると各関数従属関係が与えられた F の中のどの関数従属関係によって生成されたかがあまらかになる。

[例 4] \hat{F} は次の形であるとする。

$$\hat{F} = abc\bar{c} + acd\bar{d} + bde\bar{e} + \bar{e}a + e\bar{b}$$

$a=1, b=1, e=0$ に対応する終端節点に対応する関数は、

$$\hat{F}_{a=1, b=1, e=0} = \bar{c} + cd\bar{d} + d = 1$$

この関数を 1 にするためには 3 つの項がすべて必要であり、これらは $abc\bar{c}, acd\bar{d}, bde\bar{e}$ に対応している。すなわち、 $\{A, B\} \rightarrow E$ は $\{A, B\} \rightarrow C, \{A, C\} \rightarrow D, \{B, D\} \rightarrow E$ より導かれることを示している。

アルゴリズム 1 に対してどの項に対応する項が非終端接点で残っているかの情報を加えればよい。アルゴリズムの形式的な記述は省略する。

つぎに重要なものは、不可欠な関数従属関係を求めることである。不可欠な関数従属関係は、 F のすべての非冗長なカバーに含まれる関数従属関係である。

[アルゴリズム 3] 不可欠な関数従属関係を求める方法

- (1) F^+ のすべての全関数従属関係集合を求め、その各要素に対応する論理積をすべて求める。

(2) F に含まれるある論理積 $a_1 \dots a_k \bar{b}$ を選ぶ (不可欠なものも F に含まれる)。

(3) 上記(1)で求めた論理積のうち, $\bar{a}_1, \dots, \bar{a}_k$ や b を含まない論理積のみを選び, それらの論理和に $a_1 \dots a_k \bar{b}$ が含まれていなければ不可欠である。

最小な第3正規形を求めるには, つぎのような方法を用いればよい。

[アルゴリズム4] 第3正規形の構成

(1) $G = \emptyset$ 。

(2) F 中の不可欠な関数従属関係集合を求める。なければ(6)へ。

(3) 不可欠な関数従属関係の *key* と等価な *key* をすべて求める#。

(4) 上記(2)(3)に関する関数従属関係を G に加える。等価な *key* によって素となるために同じ関係に含みうる関数従属関係も G に加える*。

(5) F より G^+ の要素を除き(2)へ戻る。

(6) F にアルゴリズム1を適用して各々の部分木に対する関係を作りそれらの *key* の間の等価性の処理を行う#。つぎに冗長な関係の除去を行う。冗長性は他の関係で表わせることと, 他の関係を使うことにより素となる属性への

従属性*の2つの条件のもとでカバー問題を解くことになる。これをGへ加える。

(7) G中の関数従属関係より求める第3正規形を得る。

あるkeyと等価なkeyがあってもそれに対しては全関数従属にならない場合[#]や, Boyce-Coddの正規形にする場合^{*}には特別な注意が必要である。とくに $X \rightarrow A$ が不可欠でF+に $A \rightarrow B (B \in X)$ があればBoyce-Coddの正規形にすることは不可能となる。

謝辞 本論文の内容について御討論いただいた, 本学矢島脩三教授ならびに田中克己氏をはじめとする矢島研究室の諸氏に深謝します。なお本研究の一部は文部省科学研究費特定研究によるものである。また, 本研究をすすめるにあたり電子技術総合研究所の植村博士に貴重な文献のコピーをいただいたことを感謝します。

文献

- [1] E.F.Codd, "A Relational model of data for large shared data banks," CACM, vol.13, no.6, pp.377-387, June 1970.
- [2] E.F.Codd, "Further normalization of the data base relational model," in Data Base Systems, R.Rustin, Ed., Prentice-Hall, 1972
- [3] C.Delobel and R.G.Casey, "Decomposition of a data base and the theory of Boolean switching functions," IBM J. Res. Develop. vol. 17, no. 5, pp.374-386, Sept. 1972.
- [4] C.P.Wang and H.H.Wedekind, "Segment synthesis in logical data base design," IBM J. Res. Develop. vol. 19, no. 1, pp.71-77, Jan. 1975.
- [5] P.A.Bernstein, "Synthesizing third normal form relations from functional dependencies," ACM Trans. Database Systems, vol. 1, no. 4, pp.277-298, Dec. 1976.
- [6] W.W.Armstrong, "Dependency structures of data base relationships," Information Processing 74, North-Holland Pub. Co., Amsterdam, 1974, pp. 580-583.
- [7] P.A.Bernstein, "A comment on Segment synthesis in logical data base design," IBM J. Res. Develop. vol. 20, no.4, p.412, July 1976.
- [8] J.R.Slagle, C.L.Chang and R.C.T.Lee, "A new algorithm for generating prime implicants," IEEE Trans. Computers, vol.C-19, no.4, pp.304-310, April 1970.
- [9] 岡田, 上林, 矢島, "Clause Selection法による論理関数の素項生成"
電子通信学会電子計算機研究会資料, EC-74-36,
1974年11月