

# データ・ベース探索のアルゴリズム

小林 正哉

(株)日本コンピュータ総合研究所・研究所長部

データ・ベースの探索問題、つまり与えられた検索条件によつて検索を行うとこの最適なアルゴリズムを求める問題は、データ・ベース技術の中でも最も重要なものの一つであるにも拘らず、これに対する関心が比較的薄いように思われる。筆者はこの問題について一つの解を得た。

探索問題は二つの部分に大別できる。その一つは一個のデータ・ベース・ファイルに対して施される検索についての最適アルゴリズムを求めるものである。この比較的単純な問題についても探索問題はそれほど容易なものではない。実際、データ・ベースに組み入れられた索引機構の最適利用の問題を生ずるのである。検索条件が複雑になるほど探索方法も複雑になる。複雑な検索条件は例えば「文献検索システムでは日常の問題である。

第二の問題は二個以上のデータ・ベース・ファイル上の検索で、第一の問題の結果は利用するがこれと相異つた考慮を必要とする。情報代数<sup>[1]</sup>はバンドル操作の形で伝統的なデータ処理、とくにファイル・メンテナンス時のその重要性を指摘した。一方、自然言語による質問解答システムや、リレーショナル・データ・ベース・モデル<sup>[2]</sup>についての文献の多くに、その述語論理との関連が示された。また、将来データ・ベース・システムに組み入れられる可能性のある推論過程の重要な部分はこの形の検索になる。

この論文の中で用いられる用語や記法は筆者の情報空間モデル<sup>[3]</sup>のものを用いた正確な定義や使いかたについては関連論文<sup>[4][5]</sup>を参照願ひ度い。

## 探索アルゴリズム I

一個のファイルAに施される条件検索  $r[p](A) = \{x | x \in A \wedge p(x) = \text{'true'}\}$  の検索条件pは一個若しくは複數個の単位条件をつ、 $\vee$ ,  $\wedge$  の論理演算子で結合して得られる論理点肉数である。単位条件は論理基本点肉数が、二個のレンジ、コンパティブルな点肉数を関係演算子で結んだもの、あるいはこの形で分解できない論理点肉数の形をしている。もし与えられた検索条件が複數個の単位条件の論理結合であった場合、検索操作をより単純ないくつかの検索操作と付帯的な操作群と分解して使用したい。実際、より単純な検索操作をデータ・ベースの持つ索引機構を利用して効率よく実行できる可能性があるのである。

単位条件による検索はその条件の中に現れる記録属性に対して索引が用意されているか否かにより効率が異なる。ある場合には索引を利用して迅速な検索が可能であるが、ある場合にはファイルの全記録を逐一調べなければならぬ。二二で複合条件による検索の最適手段を求める問題を生ずる。

二種の単位条件が考えられる。第一種単位条件は索引の利用によつて効率よく検索が可能になるものであり、第二種は索引が役に立たないものである。ある単位条件が第一種であるが第二種であるかは必ずしも明白なものではない。索引付けの方法には直接接点によるものとインバーテッド・ファイルによるものがある。

前者は索引付けされた属性とある定数が等号で結ばれた形の単位条件に對してのみ有効である。後者は索引付けされた属性と定数を任意の関係演算子で結ばれた形のものなど多少広い応用がある。しかし乍ら、単位条件の中に唯一つの索引づけられた属性のある場合でも、これが式の左辺または右辺に単独に現われる場合以外は索引の利用はむづかしい。この形にするためには式の變形の必要があり、システムの数式処理の能力によつて第Ⅰ種であるか第Ⅱ種であるかが決まることになる。さらに単位条件の中に二個以上の属性があればたとえ全部が索引付けられているとしても組合せの問題を生ずる。この場合は余剰の場合とない限り第Ⅱ種とすべきであろう。

索引付けされていない属性を含む単位条件は当然第Ⅱ種である。第Ⅱ種の条件による検索は、検索対象となる領域の記録を一つづつ取り出して条件を満足するかどうかのテストを行うことによりねばならない。最適化は第Ⅰ種の条件を利用して第Ⅱ種条件による探索領域を狭めることが課題である。つぎに探索アルゴリズムを示す。

[第Ⅰ段] 与えられた検索条件の各単位条件が第Ⅰ種であるか第Ⅱ種であるかを決定する。もし括弧で括りきれないいくつかの単位条件の前に $\neg$ 演算子があれば、De Morgan の法則を使つて括弧を外す。 $\neg P$ の形の条件は $\neg$ とある一つの単位条件と考える。

[第Ⅱ段] 与えられた検索条件の中に第Ⅱ種単位条件と $\vee$ 演算子で結ばれた単位条件があれば(括弧で括りきれない)複合条件があれば、これを $\neg$ とある第Ⅱ種の単位条件と見做す。

[第Ⅲ段] 条件をその選言標準形に變形する。

[第Ⅳ段] 選言形の中に共通の第Ⅰ種単位条件を持つ連言項があれば、これを括弧で括り、共通な単位条件を括り出す。このやうな共通条件 $P_k$ が一個以上あれば $\{P_k\}(A)$ の元の個数の少ないものから括り出し処理を行う。 $P_k$ にインバーティド・ファイルによる索引づけが可能の場合は元の個数を実際の検索を行うことなく知ることもできる。

[第Ⅴ段] 一つの選言形中のある連言項が他の選言形の別の連言項に全部含まれている場合は前者とそれを前者に結合 $\vee$ を除去する。

[第Ⅵ段] 各連言項の中で第Ⅰ種条件を $\{P_k\}(A)$ の元の個数の少ない順に並べ、残りの第Ⅱ種条件を一纏めにして一つの第Ⅱ種条件と見做し連言項の最後に置く。

[第Ⅶ段]  $\alpha, \beta$ の二個のプライム・グリーン・スタックを用意する。

[第Ⅷ段] 第Ⅵ段で得られた検索条件を左から右に逐次調べ、つぎの規則に従つて必要な操作を実行する。

- (1) (9)の論理記号の先行なく単位条件 $P_k$ が現われる場合は $\{P_k\}(A)$ を実行しその結果に与えられた名前を $\alpha$ スタックに置く。 $\{P_k\}(A)$ の実行は第Ⅰ種と第Ⅱ種の場合で異なる。
- (2)  $\wedge$ 演算子とこれに続く第Ⅰ種単位条件 $P_k$ が現われたら、 $\alpha$ スタックから名前 $A'$ をポップ・アップし $\{P_k\}(A, A')$ を実行する。結果に与えられた名前を $\alpha$ スタックに置く。
- (3)  $\wedge$ 演算子とこれに続く第Ⅱ種単位条件 $P_k$ が現われたら、 $\alpha$ スタックから名前 $A'$ をポップ・アップし $\{P_k\}(A')$ を実行する。結果に与えられた名前を $\alpha$ スタックに置く。

- (4)  $\wedge$  演算子のつぎに左括弧が来る場合は  $\alpha$  スタックのトップにある名前を  $\beta$  スタックに写し ( $\alpha$  スタックはポップ・アップではない)、左括弧を降着する。つぎに規則 (2) または (3) が適用されることになる。
- (5)  $\vee$  演算子が現われると  $\beta$  スタックを調べ、もし  $\beta$  スタックが空でない単純に  $\vee$  演算子を降着する。  $\beta$  スタックが空でない限り、そのトップにある名前を  $\alpha$  スタックに写し、 $\vee$  演算子を  $\wedge$  演算子に変える。
- (6) 左括弧が来たり  $\beta$  スタックが空になり名前を一つポップ・アップし、括弧を降着する。

第 8 段は第 4 条件を記すが盡さるまで続けられる。ただし、(1) または (4) の規則によつて操作が行われる後に結果として得られる集合の文が十分小さくなった場合は、処理中の連立項の残りの部分も経めて第 II 種条件を充て、(3) を適用することによつて処理効率の向上は期待できる。

【第 9 段】  $\alpha$  スタックを調べ、もし二個以上の名前があれば二個の名前  $A'$  と  $A''$  をポップ・アップし、 $u(A', A'')$  を求めて結果に与え左名前を  $\alpha$  スタックに戻す。この操作は  $\alpha$  スタックに唯一の名前が残るまで続けられる。

第 9 段の終りによつて結果が得られる。第 8 段、第 9 段の操作中、第 8 段 (1) の  $P_k$  が第 II 種条件の場合と、(3) を降いて実際の記録を讀出す必要はない。検索条件を満す記録の適当な見出し (各地でよい) を保存しておけばよいのである。いや  $u$  操作はこの見出しの集合について行われる。いや  $u$  では重複元を調べる必要があるが、分類、照合操作を利用するといふ。第 9 段終了後、見出し値を使つて実際の記録が検索される。

## 探索アルゴリズム 2

つぎに二個のフアイル  $A_1, A_2$  に対する単に検索を考へる。この場合は  $R[A_1, A_2] = \{(x_1, x_2) \mid x_1 \in A_1 \wedge x_2 \in A_2 \wedge \lambda(x_1, x_2) = \text{true}\}$  を求めることになる。入は長さ 2 の論理線関数となる。探索は  $\lambda \equiv p_1 \wedge p_2 \wedge \lambda_{12}$  の形に分解することから行ひ出す。ここで  $p_1, p_2$  はそれぞれ  $A_1, A_2$  の上で定義された論理点関数であり、 $\lambda_{12}$  は論理点関数の論理結合の形に分解できない線関数である。  $p_1, p_2, \lambda_{12}$  のいずれが恒真式である場合がある。簡単のため  $R[p_1](A_1)$  を  $A'_1$ 、 $R[p_2](A_2)$  を  $A'_2$  と書くことにしよう。もし  $p_k$  が恒真式たり  $A'_k = A_k$  である。

最適探索法はつぎの三つの場合によつて異なる。

【場合 1】  $\lambda_{12}$  が恒真式たり  $R[A_1, A_2]$  は  $A'_1$  と  $A'_2$  との直積として求められる。

従つて  $A'_1, A'_2$  を構成し、実際に直積  $A'_1 \times A'_2$  を作り出す必要がある。

【場合 2】  $\lambda_{12}$  が恒真式でなくとも  $A'_1, A'_2$  が十分に小さい元を持つものであれば、 $A'_1 \times A'_2$  の元を一つづつ取り上げて条件  $\lambda_{12}$  を調べればよい。つまり  $R[A_1, A_2] = \{(x_1, x_2) \mid x_1 \in A'_1 \wedge x_2 \in A'_2 \wedge \lambda_{12}(x_1, x_2) = \text{true}\}$  の関係を用いる。

$A'_1, A'_2$  の構成には探索アルゴリズム 1 を用いることができればよい。

【場合 3】 その他の場合、つまり  $\lambda_{12}$  が恒真式でなく、かつ  $A'_1, A'_2$  の少くとも一方が多くの元を含んでいる場合にはつぎの探索アルゴリズムによるのがよい。

【第 I 段】  $\lambda_{12}$  は論理演算子で結ばれた単位条件  $\lambda_k$  で構成されているが、これらの単位条件をつぎの二種に分類する。

第 II 種:  $\lambda_k \equiv p'_1 \wedge p'_2$  の形のもの。  $p'_1, p'_2$  はそれぞれ  $A_1, A_2$  の上で定義された点関数である。

第Ⅲb 種:  $\lambda_k \text{ III } p_i' \theta p_i'$  の形のもの。  $\theta$  は第Ⅲ種以外の関係演算子。

第Ⅳ種: 第Ⅲa あるいはⅢb 種の形にならないもの。

第Ⅲa 種と第Ⅲb 種を纏めて第Ⅲ種と呼ぶ。第Ⅲ条件への分類に当って必要なら De Morgan の法則を用いて括弧を外し、第Ⅲ種であるかどうかの判定に当り可能なり関係演算子を適当に変更して第Ⅲ種を除去する。

[第2段]  $\lambda_{12}$  を選言標準形に換換する。

[第3段] 選言形の中に共通の第Ⅲ種単位条件を持つ連言項があれば、二つを括弧で括り、共通な単位条件を括り出す。

[第4段] 一つの選言形中のある連言項がその選言形の別の連言項に全部含まれている場合は、後者とそれを前者に結合し  $\vee$  を除去する。

[第5段] 各連言項中で第Ⅳ種条件を最後に集め、一纏めにし一つの第Ⅳ種単位条件と見做す。

[第6段]  $\gamma, \delta$  の二個のプロシユ・ダリッ・スタックを用意する。

[第7段] 第5段までで得られた検索条件を左から右へ逐次調べ、つぎの規則によって必要な操作を実行する。

- (1) 何の論理記号の先行もなく条件  $\lambda_k$  が現れたら、 $R[\lambda_k](A_1', A_2')$  を実行し、結果に与えられた名前を  $\alpha$  スタックに置く。
- (2)  $\wedge$  演算子とこれに続く第Ⅲ種条件  $\lambda_k$  が現れたら、 $R[\lambda_k](A_1', A_2') \cap L$  を実行する。  $L$  は  $\alpha$  スタックからポップ・アップされた名前である。結果に与えられた名前を  $\alpha$  スタックに置く。
- (3)  $\wedge$  演算子とこれに続く第Ⅳ種条件  $\lambda_k$  が現れたら、 $\{(x_1, x_2) | (x_1, x_2) \in L, \wedge_k(x_1, x_2) = \text{'true'}\}$  を構成する。  $L$  は  $\alpha$  スタックからポップ・アップされた名前である。結果に与えられた名前を  $\alpha$  スタックに置く。
- (4)  $\wedge$  演算子に続いて括弧が現れたら、 $\alpha$  スタックのトップの名前を  $\beta$  スタックに写し ( $\alpha$  スタックはポップ・アップではない)、括弧を除去する。つぎに(2)あるいは(3)が適用されることになる。
- (5)  $\vee$  演算子が来たら  $\beta$  スタックを調べ、もし  $\beta$  スタックが空なり単純に  $\vee$  を取り除く。もし  $\beta$  スタックが空でなければ、そのトップにある名前を  $\alpha$  スタックに写し、 $\vee$  演算子を  $\wedge$  演算子に変える。
- (6) 右括弧が来たら  $\beta$  スタックから名前を一つポップ・アップし、括弧を除去する。

第7段(1), (2), (3) での  $R[\lambda_k](A_1', A_2')$  の求め方は種々考えられるが、 $\lambda_k$  の種別に応じてつぎの方法が標準的なものである。

もし  $\lambda_k$  が第Ⅲa 種なら、検索は  $A_1', A_2'$  の順序照合処理が適用できる。このために  $A_1', A_2'$  の名をそれぞれ  $p_i', p_j$  を分類キーとして分類する。分類対象には少なくとも  $p_k$  と適当な記録見出しが含まれていなければならない。 $p_k$  自身が  $A_k$  の見出しである場合もある。とくに  $A_1, A_2$  の片方あるいは双方がすでにこの分類キーによって照合済に並べられている場合は前記である。 $\lambda_k$  が  $\lambda_{12}$  の中の唯一の単位条件である場合は、 $A_1', A_2'$  の片方または双方の構成の要もなく、場合に応じて  $p_i \wedge \lambda_{12}, p_i \wedge \lambda_{12}$  あるいは  $p_i \wedge p_j \wedge \lambda_{12}$  を照合条件として順序処理を行えばよい。

つぎに  $\lambda_k$  が第Ⅲb 種である場合、順序照合処理は便が悪いが一時的索引付けが利用できる。 $A_1', A_2'$  の何れか一方(元数の多い方が望ましい)を索引付け対象となる。 $A_1'$  に索引付けするにはまず  $A_1'$  の各元  $x_1$  に対して  $p_i'(x_1)$  を計算し、これを便つて  $A_1'$  の  $p_i'$  についてのインバーティッド・ファイルを作る。このインバーティッド

ド・ファイルを用い  $R[A_k](A_1, A_2) = \{(X_1, X_2) | X_2 \in A_2 \wedge X_1 \in R[P] = R_2(X_2)[A_1]\}$  の関係を用いて結果を構成することができる。  $R[P] = R_2(X_2)[A_1]$  を求める際、  $X_2$  が固定されているならば  $R_2(X_2)$  を定数ととらえることができるが、探索アルゴリズム1に帰着するのである。  $A_2$  側を索引付けする場合も同様である。 もし  $R_k$  が恒真式かつ  $A_k$  がすべて  $R_k$  について索引付けされているならば、この索引をそのまま用いることができる。一時的索引付けを省略できる。 この方法は第Ⅲa 種に力論適用できる。 第Ⅲa 種に対しては場合に応じて二つの方法のどちらかを選ぶ必要がある。

最後に第Ⅳ種の条件については、  $A_1, A_2$  の定数の多さにより場合2と同じ手段で検索を行う必要がある。

第7段処理課程である連言項処理の中間結果が十分少ない定数となったり、その項についての残りの条件は概ね第Ⅳ種条件を見做すこと。

[第8段] 単位条件と記号がすべて処理されたり、オスタックから二ファイルを減らし、Lをポップ、アポップしLULを構成して、結果に与えられた名前をオスタックに突す。この操作もオスタックに唯一の名前が残るまで終了の検索を終了する。

処理手順について探索アルゴリズム1に述べた注意はこの場合にも云える。

## 探索アルゴリズム M

一般に  $R[\lambda](A_1, A_2, \dots, A_m) = \{(X_1, X_2, \dots, X_m) | X_1 \in A_1 \wedge X_2 \in A_2 \wedge \dots \wedge X_m \in A_m \wedge \lambda(X_1, X_2, \dots, X_m) = 'true'\}$  を求める手段は E. F. Codd に基づくものと [6], F. P. Paredmo に基づくものと改変された [7]。しかしこのいずれも検索条件について精確な判別を知らせている。つぎに一般の場合についてのアルゴリズムを示そう。

[第1段] 可能な限り  $\lambda$  を  $\lambda \equiv \lambda_1 \wedge \lambda_2 \wedge \dots \wedge \lambda^{(n)}$  の形に分解する。ここで  $\lambda^{(n)}$  は  $\mathcal{A}_V = \{A_{V_1}, A_{V_2}, \dots, A_{V_{K(n)}}\} \subseteq \{A_1, A_2, \dots, A_m\}$ ,  $\mathcal{A}_{V_1} \cap \mathcal{A}_{V_2} = \emptyset$  なるファイル群  $\mathcal{A}_V$  の元の直積の上で定義された論理関数である。この場合  $R[\lambda](A_1, A_2, \dots, A_m)$  は  $\prod_{V=1}^n R[\lambda^{(V)}](A_{V_1}, A_{V_2}, \dots, A_{V_{K(V)}})$  を求め、結果の名簿中の点の順序を  $\lambda^{(n)}$  の順序に突すことにより構成される。記号の煩雑さを避けるために  $\lambda^{(n)}$  をあらためて  $A_1, A_2, \dots, A_m$  の直積の上で定義された論理関数と考へ、以下の説明を進める。

$\lambda$  は  $\lambda \equiv (\bigwedge_{k=1}^m R_k) \wedge (\bigwedge_{k=1}^{m-1} \bigwedge_{k=k+1}^m \lambda_{k_1 k_2}) \wedge (\bigwedge_{k=1}^{m-2} \bigwedge_{k=k+1}^{m-1} \bigwedge_{k_2=k+1}^m \lambda_{k_1 k_2 k_3}) \wedge \dots \wedge \lambda_{12\dots m}$  の形に単位条件分解ができる。  $R_k$  は  $A_k$  の上で定義された論理関数、  $\lambda_{k_1 k_2}, \dots, \lambda_{12\dots m}$  ( $2 \leq i \leq m$ ) は  $A_{k_1}, A_{k_2}, \dots, A_{k_i}$  の直積の上で定義された長さ  $i$  の論理関数である。これより長さの短い論理関数の論理結合の順に書けなければならないのである。

[第2段]  $A_k = R[P](A_k)$  を構成し、各々の元の数を調べて数の小さい順に並べる。簡便のためにその順が  $A_1, A_2, \dots, A_m$  であつたとしよう。

[第3段]  $L_2 = R[\lambda_{12}](A_1, A_2)$  を作る。これは探索アルゴリズム2を用いることができる。

[第4段]  $L_3 = \{(X_1, X_2, X_3) | (X_1, X_2) \in L_2 \wedge (\lambda_{13} \wedge \lambda_{23} \wedge \lambda_{123})(X_1, X_2, X_3) = 'true'\}$  を構成する。  $L_3$  の構成にあつては、  $L_3 = \{(X_1, X_2, X_3) | (X_1, X_2, X_3) \in L_2 \wedge R[\lambda_{13}](A_1, A_3) \wedge (\lambda_{23} \wedge \lambda_{123})(X_1, X_2, X_3) = 'true'\}$ , あるいは  $L_3 = \{(X_1, X_2, X_3) | (X_1, X_2, X_3) \in L_2 \wedge R[\lambda_{23}](A_2, A_3) \wedge (\lambda_{13} \wedge \lambda_{123})(X_1, X_2, X_3) = 'true'\}$  の関係を用いて交差の点に期待できる。 左に  $\otimes$  演算子  $L \subseteq A_1 \times A_2 \times \dots \times A_k \times \dots \times A_i$ ,  $L' \subseteq A_k \times A_{k+1}$  とする

二つの線集合に対し  $L \otimes L' = \{(x_1, x_2, \dots, x_j, x_{j+1}) | (x_1, x_2, \dots, x_k, \dots, x_j) \in L \wedge (x_{k+1}, x_{j+1}) \in L'\}$  を構成するものである。とくに  $\lambda_{12} \wedge \lambda_{23}$  が恒真式なり  $L_3 = L_2 \otimes \mathcal{R}[A_3, A_3']$  ( $A_2, A_3'$ ),  $\lambda_{13} \wedge \lambda_{23}$  が恒真式なり  $L_3 = L_2 \otimes \mathcal{R}[A_3, A_3']$  を得る。  
 もし  $\lambda_{12} \wedge \lambda_{13} \wedge \lambda_{23} \wedge \lambda_{123} \equiv \lambda_{12} \wedge \lambda_{13} \equiv (p_1' = p_2') \wedge (p_1' = p_3')$  かつ  $\lambda_{12} \wedge \lambda_{13} \wedge \lambda_{23} \wedge \lambda_{123} \equiv \lambda_{13} \wedge \lambda_{23} \equiv (p_1 = p_3) \wedge (p_2 = p_3)$  の形をしていれば、 $A_1, A_2, A_3$  とおき、順序照合操作を用いて第3段、第4段を経由して実行することも可能である。 $p_k$  は  $A_k$  の上で定義された論理変数関数である。  
 [第  $j+1$  段] 一般に  $L_j = \{(x_1, x_2, \dots, x_j) | (x_1, x_2, \dots, x_{j-1}) \in L_{j-1} \wedge \lambda(x_1, x_2, \dots, x_j) = \text{true}\}$  を作ることに、よって中間結果の線の長さを一つづつ伸ばすことができる。ただし  $\lambda \equiv (\lambda_{k_1}^{j-1} \wedge \lambda_{k_2}^{j-1}) \wedge (\lambda_{k_1+1}^{j-2} \wedge \lambda_{k_2+1}^{j-2} \wedge \lambda_{k_1+2}^{j-2} \wedge \lambda_{k_2+2}^{j-2}) \wedge \dots \wedge \lambda_{k_2}^{j-1}$  とする。第4段の場合と同様に  $L_j \equiv \{(x_1, x_2, \dots, x_j) | (x_1, x_2, \dots, x_{j-1}) \in L_{j-1} \otimes \mathcal{R}[\lambda_{k_1}, \lambda_{k_2}](A_k, A_k') \wedge \lambda(x_1, x_2, \dots, x_j) = \text{true}\}$  の関係を用いて効率的向上を期待できる。⑤の操作には  $\mathcal{R}[\lambda_{k_1}, \lambda_{k_2}](A_k, A_k')$  の元の  $x_k$  の見出しによる索引付けを利用するとよい。 $\lambda$  は  $\lambda$  が  $\lambda_{k_1}$  を除去したものである。また  $(\lambda_{k_1}^{j-1} \wedge \lambda_{k_2}^{j-1}) \wedge (\lambda_{k_1+1}^{j-2} \wedge \lambda_{k_2+1}^{j-2} \wedge \lambda_{k_1+2}^{j-2} \wedge \lambda_{k_2+2}^{j-2}) \wedge \dots \wedge \lambda_{k_2}^{j-1}$  がもし  $p_{k_1} = p_{k_2}$  の形の条件  $j-1$  個の論理結合である場合には、 $A_1, A_2, \dots, A_j$  の順序照合操作を用いて第3段から第  $j+1$  段までを経由して実行することも可能である。これは伝統的なパイプ処理での順序処理方式を正当化する。  
 第  $m+1$  段終了によつて  $L_m = \mathcal{R}[\lambda](A_1, A_2, \dots, A_m)$  が得られる。この操作を  $\lambda^{(V)}$  について施した後、  
 [第  $2L(V)-1+3$  段]  $L = L' \times L'' \times \dots \times L^{(V)}$  を構成し、結果の各線内の点の位置をもとの順序に戻す。

## 探索アルゴリズム M

探索アルゴリズム M 1, 2, M では検索条件に自由変数のみに関係するとしなが、ある場合には束縛変数を持つこともある。この場合検索条件は  $A_1 \times A_2 \times \dots \times A_m \times A_{m+1} \times \dots \times A_{m+q}$  の上で定義されるが結果はあくまで  $A_1 \times A_2 \times \dots \times A_m$  の部分集合である。ただし、 $A_1, A_2, \dots, A_m$  は自由変数の変域、 $A_{m+1}, A_{m+2}, \dots, A_{m+q}$  は束縛変数の変域である。束縛変数に依存する検索に対する探索アルゴリズム M はつぎのようになる。

[第1段] 条件入を連言標準形に変換し、その各項と並びに否定標準形  $Q_{k_1} Q_{k_2} \dots Q_{k_r} (\lambda^{(V)})$  の形に変換する。 $Q_{k_k}$  は  $\exists x_{m+k} \in A_{m+k}$  または  $\forall x_{m+k} \in A_{m+k}$  の形のものがあつて、もし同じ  $Q_{k_k}$  の組を持つ項があれば一纏めに1項とする。束縛変数の順序を  $Q_{k_k} = \begin{cases} \exists x_{m+k} \in A_{m+k} & 1 \leq k \leq p(V) \\ \forall x_{m+k} \in A_{m+k} & p(V)+1 \leq k \leq q(V) \end{cases}$  とおき、変更する。

この形の各項について検索結果を求めるときには、記号の煩雑さを避けるためにこれから処理すべき項が  $\lambda \equiv Q_{k_1} Q_{k_2} \dots Q_{k_r} (\lambda_E \wedge \lambda_D)$  の形であるとしよう。 $\lambda_E$  は自由変数  $x_1, x_2, \dots, x_m$  と  $\exists$  に結ばれた束縛変数  $x_{m+1}, x_{m+2}, \dots, x_{m+p}$  の変域のみに関する条件を経由したもの、 $\lambda_D$  はそれ以外の条件を経由したものである。

[第2段]  $L_E = \mathcal{R}[\lambda_E](A_1, A_2, \dots, A_m, A_{m+1}, \dots, A_{m+p})$  を構成する。

このために探索アルゴリズム M が使われる。

[第3段]  $L = \{(x_1, x_2, \dots, x_{m+1}) | (x_1, x_2, \dots, x_{m+p}) \in L_E \wedge Q_{p+1} Q_{p+2} \dots Q_{q_r} (\lambda_D(x_1, x_2, \dots, x_{m+p})) = \text{true}\}$  を構成する。

このために  $A_{m+p+1} \times A_{m+p+2} \times \dots \times A_{m+q}$  の各元  $(x_{m+p+1}, x_{m+p+2}, \dots, x_{m+q})$  について  $L_E$  の中に  $\lambda_D(x_1, x_2, \dots, x_{m+p}, x_{m+p+1}, \dots, x_{m+q}) = \text{false}$  となる元があるかどうか

かを調べ、もしそのようなものがあるならば  $(x_1, x_2, \dots, x_m, x_{m+1}, \dots, x_{m+p})$  を  $LE$  から除去するようにすればよい。  $A_{m+p+1} \times A_{m+p+2} \times \dots \times A_{m+q}$  のすべての元についてこの操作を行う。最終に  $L = \{(x_1, x_2, \dots, x_m) | (x_1, x_2, \dots, x_{m+p}) \in LE\}$  を求める。

[第  $2l+1$  段] 各入<sup>(U)</sup> に対して得られた結果の合併をとる。

探索アルゴリズム  $\Sigma$  は自由変数に束縛した条件に対してもほぼそのまま適用できる。すなわちある場合は第4段終了時に  $LE$  が空になければ「真」、空なら「偽」の解答となる。すなわちない場合は  $A_{m+p+1} \times A_{m+p+2} \times \dots \times A_{m+q}$  の全部の元が入を満足せば「真」というでなければ「偽」である。

探索アルゴリズム  $\Sigma$  1, 2, M, Q によって一階述語論理の形で与えられるすべての検索条件による検索操作に対応できるアルゴリズム  $\Sigma$  が与えられる。探索アルゴリズム 1 について2は論文<sup>[8]</sup>, 2, M, Q について2は論文<sup>[9]</sup>がある。またデータ処理への応用について2は論文<sup>[10]</sup>を参照されたい。

### 参考文献

- [1] CODASYL Development Committee. An Information Algebra: Phase I Report. Comm. ACM, Vol. 5, No. 4 pp. 190-240, Apr., 1962.
- [2] E. F. Codd, Seven Steps to RENDEZVOUS with the Casual User, IBM Research, R.J. 1333, 1974.
- [3] I. Kobayashi, An Information Space Model for Data Bases. Nippon Univac Sogo Kenkyusho, Inc., Interim Report, Oct., 1975.
- [4] I. Kobayashi, Information and Information Processing Structure. Information Systems, Vol. 1, No. 2, pp. 39-49, Pergamon Press, May, 1975.
- [5] I. Kobayashi, A Formalism of Information and Information Processing Structure; Revised Report. The Soken Kiyo, Vol. 5, No. 1, pp. 57-122, Nippon Univac Sogo Kenkyusho, Inc., Jan., 1975.
- [6] E. F. Codd, Relational Completeness of Data Sublanguages, IBM Research, R.J. 987, 1972.
- [7] F. P. Parelmo, A Data Base Search Problem, IBM Research, R.J. 1072, 1972.
- [8] I. Kobayashi, An Optimal Data Base Search Strategy for Retrievals on One File. Nippon Univac Sogo Kenkyusho, Inc., Interim Report, Nov., 1975.
- [9] I. Kobayashi, A Data Base Search Strategy for Retrievals on Two or More Files. Nippon Univac Sogo Kenkyusho, Inc., Interim Report, Dec., 1975.
- [10] I. Kobayashi, Some Enhancements on Set-theoretic Data Manipulation Languages. Nippon Univac Sogo Kenkyusho, Inc., Interim Report, Dec., 1975.