

スキーマ発見のための型近似

三 浦 孝 夫[†] 塩 谷 勇^{††}

本稿では型を近似するための発見的な方法を提案する。従来人工知能分野で知られた方法と異なるのは、データベーススキーマと実現値から構成する点にある。ある型が他の型で記述されているものを見だし、これをデータベースの見直し（再設計）にフィードバックすることが目的である。具体的には、ISA 階層に基づく近似、実現値による近似、包含に基づく近似の3つの近似方法を考察する。各方法において、データベースが本来有している仮定、つまりデータは事前に分類することができるという性質を活用し、新しい型スキーマを得ようとする。

Type Approximation for Schema Discovery

TAKAO MIURA[†] and ISAMU SHIOYA^{††}

We propose heuristic algorithms to *approximate* types semi-automatically. Unlike conventional techniques in *Artificial Intelligence*, we start from examining current schemas and the instances. We explore types that are described by other types, by which we can obtain feed-back information to database design. We propose several approaches: knowledge-based approximation, instance-based approximation and approximation by containment. In each method we will fully utilize assumptions that all the instances are classified in advance and we propose sophisticated algorithms to discover type schemas.

1. 動 機

データベース設計とは、利用者が意図することを明確にし、分析しそして定義する過程である。このフェーズは本質的に非決定的であることから、意思決定のために対話的な操作が必要となる。これまでの研究で設計や再設計のための方法を確立したとはいいがたい。

データベース設計段階では、対象世界はスキーマとして分類可能であり、その情報は問題領域とは独立に操作可能であるという仮定をおく。このことで、すべての記述や操作はデータモデルが提供する基本機能だけを組み合わせることによって実現できる。こういった枠組みはデータベースパラダイムと呼ばれる²⁰⁾。ここでは各実現値は事実に、スキーマは知識に対応すると考えることができる。たとえば、実現値の有する共通した性質を‘型’と呼び、スキーマでは型に関する記述を含むことができる。利用者やデータベースシステムが利用者の意図を検証し、所望する情報を得る方法

を最適化できるのは、この枠組みのおかげである。

しかし、この枠組みには致命的な問題がある。データベースのライフサイクルで生じる変化に動的に追従する方法がない。実際、実現値はスキーマを定義した後でなければ投入することができないため、しばしばスキーマが現在の対象世界を記述するのにふさわしくない。一般的にいうと、データベース実現値にはスキーマで失われている情報が残っている可能性が高く、そこから有用な知識（あるいはスキーマ）を得ることは妥当な方法であると考えられる。このようなアプローチをデータベースにおける知識発見 (*Knowledge Discovery in Databases, KDD*)、またはスキーマ発見と呼ぶが、本稿のねらいはここにある。

もし第1階述語論理を用いてスキーマを記述し、実現値集合を論理モデルと見なすならば、演繹的な知識を得ることは難しくない。しかし、得られる知識が演繹可能かどうかを一般には決定できないことが知られている。別の論理アプローチとして、外延（解釈）を直接用い1つ1つ調べることで、新たな知識を得ることができる可能性がある。しかし、人間とシステムとの双方向の処理の過程で膨大な数の探索が繰返し必要であり、得られた結果を対象世界の意味に翻訳することも容易ではない。

[†] 法政大学工学部

Faculty of Engineering, Hosei University

^{††} 産能大学経営情報学部

Faculty of Management and Informatics, SANNO College

本研究ではデータベーススキーマと現在の実現値集合から新たなスキーマを得る発見的な方法を提案する。本稿では、データベーススキーマのうち型のみを対象とし、1つの型の他の型による記述（近似）を試みる。この結果はデータベース設計にフィードバックされる。スキーマの検証あるいは単純化、重複や欠損の検出といった重要な情報が得られることから、これに基づいて誤ってとらえられた概念の記述変更へ寄与することができる。

すでに筆者らは、スキーマを発見し単純化するアルゴリズムを提案している^{21)~25)}。これに対して、本研究では実現値の精査手法と必要ならそれらの修正に基づく新たなパラダイムを提案する。

2章では型階層の定義と性質を述べる。3章では近似の意味を定義し、ISA階層を用いた近似手法を述べる。続く4章では、実現値に基づいた別の近似手法を導入し、さらに5章で精密化された手法を定義する。6章でこれらの特徴の要約と再設計の手順を示したあと、7章で実験結果を示す。8章で関連研究を論じる。

2. 型階層と設計問題

情報システムの構築で最も重要なものの1つは、システムの目的を理解し計算機処理に適正な問題のサイズを決定することであろう。このとき、要求が基本的であるほど強力な方法が必要となることが多い。

データモデルは、典型的には実体（entity）と連想（association）に基づいて定義される⁸⁾。前者は対象世界の‘もの’を代表する概念であり、表現方法、属性などと独立に定義される概念である。後者は実体の間の結び付きを表す概念であり、1つの結び付きは1つの連想と対応する。実体の型（type）は、実体集合に共通の性質をとらえる内包概念であり、このとき実体 e は型 t を持つという。一般的に e は t 以外にも型を持つことがある。以下では実体集合（有限とする）を $\mathcal{E}$ 、型集合（有限とする）を $\mathcal{T}$ 、型 t を持つ実体の集合を $\Gamma(t)$ と表す。これを型 t の解釈という。すべての実体を表す型が存在しないときは、これに代わる仮想的型 $\mathbf{0E}$ を考える。連想に関しても同様に考えることができるが、ここでは論じない。

各実体が型を保持するとき、実現値に基づく（instance-based）と呼び、意味モデルなど多くの例がある^{8),11)}。実体 e に対してそれが有する型を $\tau(e)$ で表す： $\tau(e) = \{t \in \mathcal{T} \mid e \in \Gamma(t)\}$ 。これを e の型スキーマ（type schema）と呼ぶ。 $\tau(e)$ は有限であるとする。すべての $\tau(e)$ はデータベースの型付け情報を表すものとなり、これをデータベースの型スキーマと

呼ぶ。

データベースがつねに正しい意味をとらえるには対象世界の変化に追従する必要があるため、型スキーマは変化せざるをえず、実体の記述や操作は複雑になるであろう。一般的にいて実体の持つ型（役割）は対象世界が複雑かつ大きくなるにつれて増加するため、 $\tau(e)$ を管理する手間は小さくない。多くの実体と大量の $\tau(e)$ 記述を個々に管理するならば、データベースの効率良い利用が困難となり、さらに一貫性制約も加われば問題がいつそう複雑になる。型階層や制約などの高度な知識制御機構を活用して、簡単化のための統一的な方法を導入し、効率良い管理機構が必要なのは明らかである。

例1 アジアにおける近隣諸国を記述するため、型 *Country* を導入し、対象とする国（実体）の型スキーマを記述しよう。ここではアジア地域を記述する8つの型を想定する：*Country*, *FarEastAsia*, *WestAsia*, *SouthEastAsia*, *Asia*, *Oceania*, *Pacific* および *Kanji*。これらの意味は自明であろう。アジア地域には14カ国が7つの型を有するものとする（以下では、すべての実体を表す型 *Country* は型スキーマから省略する）：

国	型
<i>Japan (J)</i>	<i>Asia, FarEast, Kanji, Pacific</i>
<i>Taiwan (TW)</i>	<i>Asia, FarEast, Kanji, Pacific</i>
<i>Korea (K)</i>	<i>Asia, FarEast, Kanji, Pacific</i>
<i>China (C)</i>	<i>Asia, Kanji</i>
<i>HongKong (H)</i>	<i>Asia, Kanji, Pacific, SouthEast</i>
<i>Philippines (Ph)</i>	<i>Asia, Pacific, SouthEast</i>
<i>Singapore (S)</i>	<i>Asia, Pacific, SouthEast</i>
<i>Malaysia (M)</i>	<i>Asia, Pacific, SouthEast</i>
<i>Indonesia (Is)</i>	<i>Asia, Pacific, SouthEast</i>
<i>Thailand (Th)</i>	<i>Asia, SouthEast</i>
<i>Australia (A)</i>	<i>Oceania, Pacific</i>
<i>NewZealand (NZ)</i>	<i>Oceania, Pacific</i>
<i>India (IN)</i>	<i>Asia, West</i>
<i>Pakistan (Pk)</i>	<i>Asia, West</i>

□

型を制御する典型的な機構の1つが汎化（generalization）であり、これはISAとも呼ばれる。型 t_1 , t_2 に対して、 t_2 が t_1 の汎化（または t_1 ISA t_2 ）とは、 $\Gamma(t_1) \subseteq \Gamma(t_2)$ が制約条件として与えられているときをいう。また t_1 ISA t_2 が成り立つとは、データベースにおいて当該制約条件が充足されるときをいう。ISAを複数回適用して得る反射推移閉包をISA階層（または型階層）という。この階層によってISAを正しく推論することができる。つまり t_1 ISA t_2 と t_2 ISA t_3 から t_1 ISA t_3 を導くことができる。型 t_1 , t_2 に対して t_2 が t_1 の直接汎化（direct generalization）また

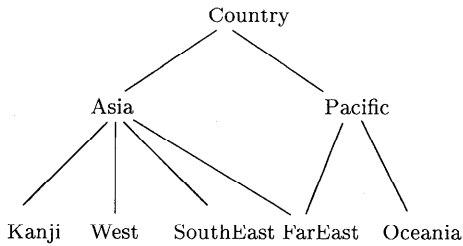


図1 Asia型の国のISA階層

Fig.1 ISA hierarchy of Asian countries.

は直接ISA, 直接親とは, t_1 ISA t_2 が成り立ち, かつ t_1 ISA t_3 で t_3 ISA t_2 となる t_3 が (ISA 階層に) 存在しないことをいう. 同様に直接子も定義できる.

複数の直接親を持つ型が存在してよいための, ISA 階層は実際には階層ではなく, 有向非巡回グラフ (DAG) であることに注意したい. にもかかわらず, 混乱のないかぎり子孫型 (部分型) や祖先型 (スーパー型) といった用語を用いる. なお, ISA は半順序となる.

実体 e が型 t を持ち, t ISA t' ならば e は t' も有するに違いない. 言い換えると, $t \in \tau(e)$ かつ t ISA t' から $t' \in \tau(e)$ が成り立たねばならない. これは型スキーマが矛盾を含まないための条件であり, 型無矛盾性 (type consistency) と呼ぶ. 以下では至るところで型スキーマの型無矛盾性が成り立つとする.

例2 例1のISA階層において9個の直接子があり, どの型もたかだか2つの直接親しか有さない. 図1は階層の一部を示している.

地域	数	国
Asia	12	C, H, In, Is, J, K, M, Ph, Pk, S, TW, Th
FarEast	3	J, K, TW
Kanji	5	C, H, J, K, TW
Oceania	2	A, Nz
Pacific	10	A, H, Is, J, K, M, Nz, Ph, S, TW
SouthEast	6	H, Is, M, Ph, S, Th
West	2	In, Pk

意味データモデルの多くは汎化, 専化, 集約などを含んでいる⁸⁾. 一般的にいて, 一貫性制約, 演繹規則やISA階層などの知識間には相互に関連があることが多く, たとえば型の上下概念という階層関係はコンパクトな知識表現を作成するうえでとくに重要である. しかしこれらは系統だって理解されないことが多く, 設計 (知識の獲得) が大きい問題である¹⁵⁾. しかし, その設計はこれまで知られた手法や知識獲得技術を用いたとしても, 手作業に頼らざるをえない³⁴⁾. 実際, 型の階層に関しては系統だった設計手法は知られておらず, 経験的な知識や思いつくままに列挙する以外に手がない.

データベーススキーマとして定義される知識は, 管

理された分類方式として利用される. この方式に従えば, 各情報ごとに '概念' 情報を特定され, あらかじめ定められた方式で格納管理される. これに対して, 事前の分類なしに格納された情報は無造作のまま放置されているため, 何らかの類似点や差異を抽出することでクラスタ化され, '概念形成' に至る¹⁸⁾. データベースからの概念形成 (データベースからの知識獲得) は, より高度で精密な分野への適用がなされるにつれ, 中心的で難しい問題となりつつある.

データベースから得た知識は, 検査され有用であることを確認したのち, データベースに戻される. しかし, これまで知識獲得技術として知られた手法は, 指数オーダの実行時間を要するアルゴリズムが多く, 効率の観点からそのまま適用することが難しい. したがって, 現実には差分的な知識獲得や問題領域知識を活用する方式が妥当である. いったん設計された型情報やISA階層を用いて, 対象世界の変化を差分情報からスキーマとして記述しなおす過程を, スキーマの再設計 (redesign) と呼ぶ. 本研究はISA階層および実現値それ自体に対する再設計手法を示すものである.

3. 型の近似

1章で述べたように, データベーススキーマをより良いものにするための工夫には様々なものがある. ここでは1つの型が他で近似されているものを見いだす問題を論じる. 型 t の $t_1, \dots, t_n$ による近似 (approximation) とは, $\Gamma(t)$ が $\Gamma(t_1), \dots, \Gamma(t_n)$ によって 'おおよそ記述される' ことを意味する. ここで問題となるのは, '記述' や 'おおよそ' という言葉の明確な定義と近似の方法である.

型近似が意味するものを理解するために, 初めにISA階層と型近似の関係を論じよう. 型 t に対して, T_t をその子孫集合とする: $T_t = \{s \mid s \text{ ISA } t\}$. 一般に $s \in T_t$ に対して $\Gamma(s) \subseteq \Gamma(t)$ である. 特に $\bigcup_{s \in T_t} \Gamma(s)$ が $\Gamma(t)$ に等しいとき, T_t は t の被覆 (covering) である, あるいは T_t は t を被覆するという. このとき t は T_t から和操作によって得られるから一種のビューと見なせる. 多くの文献^{7), 11), 16), 17)} でISAと被覆は独立した概念として論じられているが, しばしば t の近似を t の被覆と考えることができる.

$T_t = \{t_1, \dots, t_n\}$ が t を被覆し t_1 ISA t_2 であるとき, $T_t - \{t_1\} = \{t_2, \dots, t_n\}$ は依然として t を被覆する. T_t の部分集合 T_1 が t を被覆するとき, (現在論じている) ISA階層に関して T_1 が非冗長であるとは, T_1 のどの真部分集合も t を被覆できないときをいう. $T_t = \{t_1, \dots, t_n\}$ が t を被覆するならば, t に

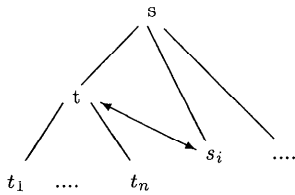


図2 ISA 階層の補助的な型

Fig. 2 Supplementary types in ISA hierarchy.

直接 ISA となる t' の集合は明らかに非冗長となる。

T_t が t を被覆しないとす。データベースにおいてはしばしば t が他の (子孫ではない) 型 $s_1, \dots, s_m$ によって被覆されることがある。たとえば、スキーマでは記述されていないにもかかわらず、業務知識を有する人物が個々の実現値を操作するならば、暗黙のうちに意味を保全し続けることがある。このことは、データベース設計時に ISA 条件を指定し忘れた可能性を示唆している。つまり、 $\Gamma(s_i) \subseteq \Gamma(t)$ が成り立つならば s_i ISA t である可能性を否定できない。むしろ、偶然に $\Gamma(s_i) \subseteq \Gamma(t)$ となることもあるからこの決定は断定的では有りえない。データベース管理者や設計者が真に ISA なのかどうかを確認する必要がある。問題は、どのようにしてこういった‘補助的な’型を見いだすかにある。膨大な型の集まりから、このような検査を手作業で行うのは明らかに現実的ではない。

本稿で提案する最初の発見の方法は、ISA 階層を用いて候補を絞るというアイデアに基づく。型 t の祖先型 s の解釈は T_t の各型の解釈をすべて含み、さらに候補の型についても解釈を含んでいる。したがって、 s の (t , T_t 以外の) 子孫型を調べ、 t に近似ができないかどうかを検査する。図2はこの状況を表している。型 s_t は実は t の子型である可能性がある。

このアイデアは、ISA 階層の一部が正しくモデル化されていないが他の部分が依然として有効である場合に、型を近似できる能力を持つ。おそらくデータベース管理者はこの制約を評価し真の ISA かどうかを判定するであろう。次の手順は、補助的な型を検出するのに有用である。

手順は候補型の生成を行いながら進行する。 t を近似したい型、 T_t を t のすべての子孫型から冗長な要素を取り除いた集合、 V を T_t の要素の解釈の和とする。初めに t のすべての直接親 T_a を計算し、各要素 $s \in T_a$ に対してその直接子 c で t でないものを選ぶ。もし $\Gamma(c) \subseteq \Gamma(t)$ ならば c は候補である。実際 $\Gamma(c) \cap V = \phi$ ならば c は解の1つであり、 c を T_t に、 $\Gamma(c)$ を V に加える。 $\Gamma(c) \subseteq \Gamma(t)$, $\Gamma(c) \cap V \neq \phi$ かつ $\Gamma(c) \not\subseteq V$ のときも、 c を T_t に、 $\Gamma(c)$ を V に加

える。 $\Gamma(c) \not\subseteq \Gamma(t)$, $\Gamma(c) \cap V \neq \phi$ かつ $\Gamma(c) \not\subseteq V$ ならば、 c の子孫の一部が解であることが考えられるため、 c のすべての直接子を新たな候補としてこの処理を繰り返す。こうして得られた結果 T_t は、 $\Gamma(c) \subseteq V$ のときは解に加えないから、非冗長である。

$V \subseteq \Gamma(t)$ で $\frac{|V|}{|\Gamma(t)|} \geq \alpha$ ならば、 t は T_t で近似されている。ここで $0.0 < \alpha \leq 1.0$ である。 $\alpha = 1.0$ とは $V = \Gamma(t)$ を意味し、 α は近似の度合い (しきい値) と考えられる。 $\frac{|V|}{|\Gamma(t)|}$ を t の近似率 (approximation ratio) という。実際にしきい値を満たす近似率を実近率という。以下は手順の要約である。

- (1) t を型、 T_t をそのすべての子孫型で冗長なものを取り除いた集合、 V を T_t の要素の解釈の和とする。
- (2) T_a を t のすべての直接親リストとする。
- (3) 各 $s \in T_a$ に対して (3.1) から (3.2) まですを繰り返す
 - (3.1) C_s を s の直接子リストとする。
 - (3.2) 各 $c \in C_s$ で t と異なり T_t に含まれないものが存在する限り、(3.2.1) から (3.2.5) まですを繰り返す
 - (3.2.1) $\Gamma(c) \subseteq \Gamma(t)$ かつ $\Gamma(c) \cap V = \phi$ ならば、 c を T_t に、 $\Gamma(c)$ を V に加える。
 - (3.2.2) $\Gamma(c) \subseteq \Gamma(t)$ かつ $\Gamma(c) \not\subseteq V$ ならば、 c を T_t に、 $\Gamma(c)$ を V に加える。
 - (3.2.3) $\Gamma(c) \not\subseteq \Gamma(t)$ かつ $\Gamma(c) \not\subseteq V$ ならば、 c のすべての直接子を C_s に加える。
 - (3.2.4) いずれでもなければ何もしない。
 - (3.2.5) $V \subseteq \Gamma(t)$ かつ $\frac{|V|}{|\Gamma(t)|} \geq \alpha$ ならば、 t は T_t によって実近率 $\frac{|V|}{|\Gamma(t)|}$ で近似されており終了する。

型数 m , 実体数 n とし、ISA 階層をリストの配列 `parent`, `child` で表現する: t ISA s のとき `parent[t]` に s が、`child[s]` に t が含まれるとする。 T_t を型の (何かの順序による) 昇順リストで、また各型 t に対して $\Gamma(t)$ を実体の (何かの順序による) 昇順リストで表す。このとき、(1) は $O(nm)$, (3.1) は $O(m)$, (3.2) は $O(m)$, (3.2.1), (3.2.2) の各ステップは $O(n+m)$, (3.2.3) では C_s を作り直すので $O(n+m)$, (3.2.5) は $O(n)$ の手間が必要となるから、(3), (3.1) での最大 m 回のそれぞれの繰り返しを考慮すれば、全体の計算量は $O(nm^2 + m^3)$, つまり m, n に関して多項式時間で済む。

例3 例1と例2において、Asia は Kanji, West, SouthEast および FarEast によって近似される、このとき近似率 α は 1.0 であってもよい。この結果は、Asia の解釈が ISA 階層による意味を正しく保持していることを検証したことになる。あるいは、ISA 階層から除去し、Asia を計算で求めるとし、ISA 階層を簡単にするという判断もできる。 $\alpha = 0.50$ とした場合、

Pacific は *FarEast* と *Oceania* で近似できる。この場合実近似率は 0.50 であるため、この近似は正しくないと感じるかもしれない。□

上述の手順では t は結果として t の子孫またはその候補で記述され、 t の解釈はそれらの和で表される。このような近似を内部近似 (*interior approximation*) と呼ぶ。逆方向に考えれば外部近似することができる。すなわち型 t を先祖またはその候補で近似し、 t の解釈をそれらの共通部で記述すればよい。

4. 実現値に基づく型近似

前章では、発見的な方法により ISA 階層を用いた近似方法を論じた。ISA 階層の持つ推移性や継承機能により、この方法は知識に基づく近似の 1 つであるが、これが効率的かつ効果的であるのは、階層の大半が対象世界とデータベース実現値の対応を正しく表しているときに限る。階層が実状とかけ離れているならば、手がかりをえることさえ期待できず、この場合新たな方法が必要となる。

すべての実現値が対象世界の情報を正しく保持しているとすれば、これらを個々に検査し型スキーマを用いた近似を行う必要がある。先の方法と比較すれば、この方法はエレガントでも高速でもないが、実状に即した結果を得ることができる可能性がある。このような方法を実現値に基づく近似 (*instance-based approximation*) と呼ぶ。

第 2 の発見的方法のアイデアは、型 t の各実体 $e \in \Gamma(t)$ が持つ型スキーマから出発する。 T_t を型 s で、(1) ある $e \in \Gamma(t)$ に対して $s \in \tau(e)$ であり、(2) $\Gamma(s) \subseteq \Gamma(t)$ とをみたすものの集合とする： $T_t = \{s | e \in \Gamma(t), s \in \tau(e), \Gamma(s) \subseteq \Gamma(t)\}$ 。 T_t の要素の解釈のすべての和は $\Gamma(t)$ の部分集合であり、 T_t は t を被覆する可能性がある。

新たな近似の手順は次のとおりである。 T_t を $\Gamma(t)$ の要素の持つ型スキーマの和とし、 U 、 T を空集合とする。各 $r \in T_t$ で t でないものに対し、 $\Gamma(r) \subseteq \Gamma(t)$ でしかも $\Gamma(r) \not\subseteq U$ かどうかを検査する。もしこれが成り立てば、 r は解の 1 つであり r を T に加え、 $\Gamma(r)$ を U に加える。 U はつねに $\Gamma(t)$ に含まれることに注意したい。先と同様に、 $\frac{|U|}{|\Gamma(t)|} \geq \alpha$ が成り立てば、近似を完了する。ここでも得られた結果 T は、 $\Gamma(r) \subseteq U$ となる r は取り除かれるから、非冗長である。以下に示すのはこの要約である。

- (1) t を型、 T_t を $\Gamma(t)$ の実体の型スキーマの和、 U 、 T を空集合とする。
- (2) 各 $r \in T_t$ で $t, @E$ でないものに対して、(2.1)

から (2.2) を繰り返す。

(2.1) $\Gamma(r) \subseteq \Gamma(t)$ かつ $\Gamma(r) \not\subseteq U$ が成り立てば、 r を T に、 $\Gamma(r)$ を U に加える。

(2.2) $\frac{|U|}{|\Gamma(t)|} \geq \alpha$ が成り立てば近似を完了する。

型数 m 、実体数 n とし、 T 、 T_t を型の (何かの順序による) 昇順リストで、また U および各型 t に対して $\Gamma(t)$ を実体の (何かの順序による) 昇順リストで表す。このとき、(1) は $O(mn)$ 、(2.1) は集合の比較に $O(n)$ 、合併に $O(m+n)$ 必要とし、(2.2) は $O(n)$ でよいから、(2) での最大 m 回の繰返しを考慮すれば、全体の計算量は $O(mn + m^2)$ 、つまり m 、 n に関して多項式時間ですむ。

この実現値に基づく近似も内部近似の一種であり、前の方法と同様に、実現値に基づく外部近似を定義することができる。実際 T_t の定義において、型 s をある $e \in \Gamma(t)$ が型 s を持ち $\Gamma(s) \supseteq \Gamma(t)$ となるようにすればよい。

例 4 例 1 と例 2 において、*Asia* は *Kanji*、*West*、*SouthEast*、*FarEast* によって近似される (近似率 α は 1.0)。 $\alpha = 0.60$ の場合、*Kanji* は *FarEast* だけで近似される (実近似率 0.60)。また *Asia* は *Kanji*、*FarEast*、*SouthEast* でも近似される (実近似率 0.83)。この結果は 1 つあるいは 2 つの型が他の型でおおよそ記述され、実際にその意味を検証するほうがよいことを表している。もし、我々がこれらの概念定義を変更するならば、単に ISA だけではなくて (述語や連想などの) 関連する他の情報も変更せねばならない。□

ISA に基づく近似と比較した場合、この方法はいかなる知識も利用せず実現値だけを用いたことから、解に至るまでに時間を要することが予想できる。しかし、その結果は現在の状態を反映しており、再考察のためには多くのヒントを得るであろうことが期待される。

5. 包含に基づく型近似

これまで論じてきた近似方法は、得られた近似結果を新たな ISA としてフィードバックするものであった。もし近似度が 1.0 ならば、実現値に対する修正はまったく起きないことに注意したい。

本来、近似という用語は、2 つの表現の意味がある観点から同一になることを意味し、必ずしも論理帰結を意味しない。ある概念を他で近似したとき、一貫性を保つために関連情報を修正したり見直す必要が生じる。以下では、もはや内部あるいは外部近似にこだわらず、実現値の調整を考える。

いくつかの型が表す和概念を '正しい解' S_1 と定め、

それを1つの型で表現できる範囲 S_2 と比べれば, 型近似を型近似を以下で定義する外接率と内接率で表現できそうである.

外接率を $\frac{|S_2|}{|S_2 \cap S_1|}$, 内接率を $\frac{|S_1 \cap S_2|}{|S_1|}$ と定義し, 実現値とその情報(型スキーマ)を用い, どのような解釈が基準となる内接外接率を達成するかを考える*. 内接率は残存する誤り率を, また外接率は処理すべき情報がまだ残っていることを意味し, とともに1.0になるほど近似率が高い. このような方法を包含に基づく近似 (approximation by containment) と呼ぶ.

提案する手順はこれまでのものと似ている. T_t を型 t の実体の型スキーマの和とし, また U, T を空集合とする. 各 $r \in T_t$ で $t, @E$ ではないものとするれば, $\Gamma(r) \subseteq \Gamma(t)$ かつ $\Gamma(r) \not\subseteq U$ かどうかを検査する. もしそうなら r は解であり, これを T に加え, また $\Gamma(r)$ を U に加える. $\Gamma(r) \subseteq \Gamma(t)$ と $\Gamma(r) \subseteq U$ の両方が成り立てば, r の解釈はすでに含まれている.

$V = \Gamma(r) \cup U$ とする. $\Gamma(r) \not\subseteq \Gamma(t)$ であるとき, 外接率 $\frac{|V|}{|V \cap \Gamma(t)|}$ が α_1 より小さいかどうかを調べ, もしそうなら r を T に, $\Gamma(r)$ を U に加える. このとき, 近似率 α_1 の範囲で誤ったデータが混入される可能性があり, α_1 は再現性に関するしきい値となっている.

この処理は U は α_1 の範囲で増加し, 内接率 $\frac{|V \cap \Gamma(t)|}{|\Gamma(t)|}$ が α_2 より大きくなるまで繰り返す. α_2 は適合性に対するしきい値として働く. ここでも, 得られた結果 T は非冗長である. 以下が提案する手順である.

- (1) t を型, T_t を $\Gamma(t)$ の実体の型スキーマの和, U, T を空集合とする.
- (2) 各 $r \in T_t$ で $t, @E$ でないものとするとき, (2.1) から (2.3) までを繰り返す.
- (2.1) $\Gamma(r) \subseteq \Gamma(t)$ かつ $\Gamma(r) \not\subseteq U$ が成り立てば, r を T に, $\Gamma(r)$ を U に加える.
- (2.2) $\Gamma(r) \not\subseteq \Gamma(t)$ かつ $\Gamma(r) \not\subseteq U$ であり, $\frac{|V|}{|V \cap \Gamma(t)|} \leq \alpha_1$ ならば, r を T に, $\Gamma(r)$ を U に加える. ここで, $V = \Gamma(r) \cup U$ とする.
- (2.3) $\frac{|V \cap \Gamma(t)|}{|\Gamma(t)|} \geq \alpha_2$ ならば処理を終了する.

型数 m , 実体数 n とし, T, T_t を型の (何かの順序による) 昇順リストで, また U および各型 t に対して $\Gamma(t)$ を実体の (何かの順序による) 昇順リストで表す. このとき, (1) は $O(mn)$, (2.1), (2.2) は集合の比較に $O(n)$, 合併に $O(m+n)$ 必要とし, (2.3) は $O(n)$ でよいから, (2) での最大 m 回の繰返しを考慮すれば, 全体の計算量は $O(mn + m^2)$, つまり

m, n に関して多項式時間でよい.

これまでの方法と違って, この方法は内部近似でも外部近似でもなく, 2つのしきい値を用いて近似の上下限を与えたものとなっている.

例5 内接率, 外接率をともに1.00とすれば *Asia* は *FarEast*, *Kanji*, *SouthEast*, *West* で近似される. しかし, 外接率を1.15, 内接率を0.85とすれば, *Pacific* は *FarEast*, *Oceania*, *SouthEast* で近似でき (このとき実外接率は1.10, 実内接率は1.00), これら3つの型が *Pacific* を表せるが完全ではないことが分かる. この例外となる実体は *Th* であり, これは *SouthEast* 型を有するが *Pacific* を有さない. 他の *SoutheEast* の国々が型 *Pacific* を有するのに比べて特徴的である. この場合, 本当の解は *Thailand* が本当に *Pacific* かどうかを考えたのでは得られない,むしろ *Pacific* が何を意味するかを判断すべきである. たとえば, *Thailand* が *Pacific* であると決断された場合 (つまり実現値を修正してこの妥当性を認めた場合), *Pacific* は *FarEast*, *Oceania*, *SouthEast* によって完全に近似され, その概念は *AsianMarine* と呼ぶべきものに変化する. □

実現値に基づくアプローチと比較した場合, 何も知識制御せずに内部的かつ外部的に被覆することから, 実体集合の探索範囲が広がり, 解に至る時間がさらに必要となる. ただ, 実体集合の正確な包含性を検査しなくなったため, 得られる結果は, 型の持つ意味を見直さねばならないような内容を含む可能性がある.

6. 考 察

本稿で提案した近似結果を評価するために, データベーススキーマについていくつかの基準を論じ, データベースの見直しのための指針を示す.

‘被覆状態 (Covering status)’とは, ある型が子孫型だけで完全に近似されている状況をいう. この場合 *ISA* は正しく効果的に設計されており, しかも当該型は1種のビューと見なせる状況を意味している可能性が高い. 特に‘異音同義状態 (homonym status)’はただ1つの型だけで被覆されているときをいう. もし‘直接子が1つしかない型’ビューという意味が意図されたものでなければ, 同じものが異なって定義されている可能性がある.

‘非被覆状態 (Non-covering status)’とは, ある型が子孫型で近似されているが被覆ではない状況をいう. この場合, *ISA* はおおむね正しくまた効果的に作用しているが完全ではなく, 子孫型だけでは被覆されない実体が存在している. 当該型は近似できるため, 解釈

* 情報検索分野では, 適合率 (precision) $\frac{|S_2 \cap S_1|}{|S_2|}$ と再現率 (recall) $\frac{|S_1 \cap S_2|}{|S_1|}$ の2つの基準を用いるが, 本稿との直接の関連はない.

のほとんどは子孫によって被覆できる。関連する実体の型スキーマを修正すれば完全な近似を得られることから、データベース設計者あるいは管理者に対して、型の意味を変更する手がかりを与えるものとなる。

‘欠落状態 (Missing status)’とは、ある型が子孫型および非子孫型によって完全に被覆できる状況をいう。この場合、ISAは正しく作用しているが他の型と組み合わせれば被覆できている。おそらく、あるISA定義が欠落しており、このことを重要なヒントとしてデータベーススキーマを見直すべきであろう。とくに、欠落状況での異音意義型が自明でない場合^{*}、重複定義になっているかどうかを検討するヒントを与える。

‘再検討状態 (Re-visiting status)’とは、ある型が子孫型および非子孫型によって近似できるが被覆ではない状況をいう。この場合、現在のISA階層とは独立に近似できることから、データベース設計の結果が対象世界の興味有る側面をうまくモデル化したとはいにくい状況にあることを示しており、何らかの混乱を含む可能性が高い。近似できるという事実から、当該型の有する概念を見直して（近似構成の型を用いて）もっと正確に論じるべき可能性がある。

このような考察から、型近似の結果はデータベース設計に重要で本質的な情報を提供することができる。これらを利用して、既存データベーススキーマ（ここではISA階層）および個々の実体に関する型スキーマの見直しを行うことができる。

ISA階層の妥当性の検証：

更新の比較的少ない状況（頻繁な定期検査あるいは記録の解析などによって判断する）では、現データベーススキーマは‘原型をとどめている’可能性が高い。この場合、ISA階層を用いた近似により被覆状態の確認（被覆すべき状況の確認）、欠落状態や再検討状態の型検出による状況の判定を行う（予期せぬISAの検出）。欠落状態や再検討状態の型検出では初期設計の誤りかどうかを検討する。結果として新たなISAの追加判定が行われる。

更新が激しいか多い状況（少ない頻度での定期検査あるいは記録の解析などによって判断する）では、実現値に基づく近似あるいは包含に基づく近似を用いる。被覆状態の確認、欠落状態や再検討状態の検出を行い、ISA階層の変更を検証する。この作業によって、データベースの変化による新たな意味がISAの検出（被覆状態の有無や欠落状態の判定）によって得られ

たことになり、また型 t が $t_1, \dots, t_n$ による近似から $s_1, \dots, s_m$ によるものに再定義されるべきという変更を行う手がかりを得たことになる。これらの結果から、被覆状態の判断（被覆すべき状況の確認）、ISAの追加や定義変更を行う。

重複定義の検出：

いずれの近似アルゴリズムを用いたとしても、欠落状況にある異音同義型が重複して定義されていないかどうかを確認すべきである。一般的にいて、異なる型が同じに定義されている状態に比べて、同一型が異なって定義されている状況を見いだすのは容易ではないことに注意したい。

ISA階層の簡単化：

被覆状態にある型はISA階層を介して正しく管理されている。ISA階層を簡略化する必要があるときは、当該型を取り去ってよい。質問式での言及を近似型で置き換えても、いずれの近似アルゴリズムによっても動作の保証がなされている。

実現値修正による見直し：

近似状態で表現されている型を実現値修正によって見直すことができる。実際、非被覆状況や再検討状態にある型 t は‘おおむね’型集合 $t_1, \dots, t_n$ によって記述されている。このとき（実現値を見ることなく） t の意図を $t_1, \dots, t_n$ によって表せ、対象世界の意味（あるいは意味の変化）を抽出した可能性を検討できる。これを特定するために、例外となる実体（つまり $\Gamma(t) - \bigcup \Gamma(t_i)$ の要素）を個別に検査し、型つけを取り除くか、あるいはいずれかの t_i に‘転向’可能かどうかを決定する。 t という型分類を残存させる意義を認めるかどうかは設計者あるいは管理者によるが、この近似作業によって、どの実体をどう修正すればよいという基準を得ることになる。非被覆状況の見直しとは、実はこの近似は被覆状況の可能性を判定することであり、再検討状況による見直しとは、関連する実体の修正で型の持つ意味を再定義できるかどうかを判定することになる。

3つの近似アルゴリズムのいずれを用いても、実現値修正によるデータベース見直しの意義は変わらない。更新の比較的少ない状況では、ISA階層を用いた近似により非被覆状態（被覆可能な候補の抽出）や再検討状態（この場合は型の定義見直しをとまなう）が、初期設計の誤りとして検出されやすい。更新が激しいか多い状況で実現値に基づく近似あるいは包含に基づく近似を用いた場合、非被覆状態の確認や再検討状態の検出により、現在の対象世界の意味変更の確認を検証できる。

^{*} t ISA s となる型 t , s の解釈が一致するときは、近似手順では s ISA t という自明の欠落状況を得る。

7. 実 験

この章では近似アルゴリズムの有効性に関する実験結果を示す。データベースの再設計と近似アルゴリズムとの関連を直接評価することは、他のリバースエンジニアリング分野と同様簡単ではない。というのは実データとして何を用いればよいのかという問いに汎用的に答えることが難しく、また問題領域固有の知識の評価をともなうからである。そこで本稿では、雑誌の記事とそれにつけられたキーワード索引を用い、近似アルゴリズムとデータベース再設計との連携とその有用性を傍証する。このようなアプローチをとった理由として、実際のデータが時代とともに変化していく様子が理解しやすく、しかもスキーマ (ISA 階層) の意味や意義が変化につれて評価され直す状況をみてとれるからである。

7.1 準 備

ここで用いるデータは、1985年から10年間分の日経バイト誌のキーワード索引であり、合計2315個の記事 (このうちタイトルが間違っ2件挿入されており、これを除外する) が対象となる。各記事レコードは、タイトル、出現年月、巻番号、開始ページとキーワードリストを含む。以下では各レコードを型ARTICLEの実体と見なす。さらに「キーワードを型と見なす」ことで各記事は複数の型を有する*。索引には196個のキーワードが定義されており、そのうち実際には146個が延べ2792回出現している**。以下は索引の例である：

- (488) GUI based DTP softwares on Unix/Windows:
1992.04:no.98:p.148:Product Report:DTP:
GUI:UNIX:Windows
- (580) Demonstrating by IBM and MS - OS,
Multimedia, Pen:1991.12:no.93:p.144:Trends:
OS Trends:Multimedia:Pen Computer

この索引で、たとえば488番目の項目のタイトルはGUI based DTP softwares on Unix/Windowsであり1992.04の98号に148ページ目から出現している。この記事はProduct Report (製品レポート) として分類されており、4つのキーワードDTP, GUI, UNIX, Windowsを有する。

この索引ではISA階層が288個の (直接親との) 組<キーワード, 直接親>として記述されており、@Eが直接親となる汎化が53組ある。全体として325個の

枝が5レベルに分岐して記述され、各キーワードはたかだか5つの直接親しか有さない。次は汎化の一部を示している：

AI:@E
APL:Language
AX:IBM Compatible:Personal Computer
Ada:Language
Apple II:Apple Product:Personal Computer
Apple Product:@E

ここでは、たとえばApple II ISA Apple Productであり、Apple II ISA Personal Computerでもある。またApple Product ISA @Eが成り立っている。

次は索引の要約である：

キーワード	記事数
1	304
2	44
3	422
4	536
5	341
6	368
7	150
8	75
9	48
10	13
11	9
12	1
13	2
合計	2313
平均	4.35 記事あたり

親	数
1	112
2	77
3	6
4	1
合計	196 キーワード
	288ISA
平均	1.47 キーワードあたり

7.2 ISAに基づく近似

以下ではSun SparcStation20/61を用いてSunOS-4.1.3の上にGNU AWKによって実験された。

初めにISAに基づく近似を、近似率0.90で実施した。この要約を示す：

状態	対象型数
被覆	47
非被覆	1
欠落	3
再考察	0
合計	51
異音同義 (被覆)	16

この実験では多数の被覆状態が検出された。これらはISA階層の定義と正確に連動しており、矛盾のない意味を保持している。たとえば、AppleIIとMacintoshはApple Productを構成している。

ISA階層の妥当性の検証については、欠落状態にあった3つの型によってなされた。これらの型はIBM Compatible, Personal ComputerおよびOSであり、欠落している条件は次の3つであり、これらは (いまとなってみれば) おおむね妥当な内容である。

PC-AT Compatible ISA IBM Compatible
Windows-NT ISA Personal Computer

* さらにProduct Report, Trends等の分類も与えられているが、あらかじめ用意した数種類のいずれかであり、キーワードの持つ多様さと比較しにくい。このため、この実験では考察から除外した。

** 処理のしやすさから索引を英語に変換してある。

Windows-NT ISA OS

この3つのISAはISA階層に追加されるべきものと判断されれば、ISA階層を修正し上述の3つを追加して欠落状態を取り除く。

重複定義の検出については、被覆状態ではない異音同義型は検出されなかったため、(つまりすべてがISA階層内部で定義されているため)このままで問題はない。参考までに、被覆状態にある異音同義の型をすべて列挙する。

ReadOnly	CD-ROM
Epson Product	PC-286/386/486..
Sharp Product	X68000
Data Compression	Disk Compression
Hard Disk	Disk Compression
File System	File Management Software
Memory	Semi-Conductor Disk
User Interface	GUI
OKI Product	11800
Voice	Voice Synthesis
Description	Language
Magnetic Tape	Streamer
Performance Evaluation	Bench-Mark
TOSHIBA Product	J-3100
Japanese Processing	Front-End-Processing
HITACHI Product	B16/32

右辺の型は左辺の子孫と定義されていた。これ以降の近似処理でも同解釈を有することから何度も生じる。

実現値修正によるISA階層の見直しについては、非被覆状態にあるただ1つの型だけが候補となった。該当する型はDatabaseであり、これは実近似率0.924でDatabase Management Softwareに非被覆的に近似できている。このことは、Databaseの型を持つ記事の92.4パーセントは同時にDatabase Management Softwareをキーワードに有しており、残りの記事はデータベース設計や処理効率などを論じていることを意味する。本実験でのDatabaseに関する記事は85件あり近似度が高いものの、たとえばデータベース設計や処理効率とDatabase Management Softwareを同一視することは困難であると判断されれば、このまま変更しなくてよい。

7.3 実現値に基づく近似

実現値に基づく近似も同じ近似率0.90を指定して行った。この実験結果は次のとおりである：

状態	対象型数
被覆	35
非被覆	7
欠落	18
再考察	5
合計	65
異音同義(被覆)	13
異音同義(他)	14

ISA階層の妥当性の検証を行うために、欠落状態の型

を分析した結果、3つの新たな型Word Processing, Development, Languageを得た。これは前の方法では得られなかったものである。見落としと思われるISA条件は次の3つであり、これらは(現時点では)いずれも妥当である：

Japanese Processing ISA Word Processing
Description ISA Development
Description ISA Language

妥当と見なすならばISA階層に追加修正する。異音同義となる型については、ISAに基づく近似とまったく同じ結果が得られた。

この近似アルゴリズムでは、前のものと違って非被覆状態の型の数が7に増えた。たとえば、型CPUはRISC, Add-on CPU Board, Graphics Processors, Co-Processors, Micro-Processors, TRONの6つによって完全に近似されたが、同時にまた近似率0.911でRISC, Add-on CPU Board, Graphics Processors, Co-Processors, Micro-Processorsの5つだけでも近似できた。言い換えると、型TRONの効果はCPUを決定するほど大きくはなかったことになる。もしこの近似を認めるならば、型CPUの名前(とその意味)をいっせいに変更してFashionableCPUとし、12件のTRON記事を検査してキーワードFashionableCPUを取り去り、ISA階層からTRON ISA FashionableCPUを除外する。

重要な差は再考察状態の5つの型で見られた。実際OS, Software, Network, Personal Computer, Storage Deviceの5つの型はISAに基づく近似では被覆状態の型と見なしたものである。この近似手法では、非子孫型を同時に考察することで0.90以上の実近似率による近似が可能であることが判明した。言い換えればISA階層の当該部はうまくモデル化できていないように思われる。たとえば、ISAを用いた近似ではStorage Deviceは10個の子孫型CD-ROM, IC Memory Card, Streamer, Hard Disk, Floppy Disk, Memory Board, Memory Architecture, Optical Disk, Semi-Conductor Disk, Digital Notebookで完全に近似された。しかしこの近似アルゴリズムでは、9つの型CD-ROM, Streamer, Hard Disk, Floppy Disk, Memory Board, Memory Architecture, Optical Disk, Memory, Digital Notebook(非子孫型はMemoryのみ)で近似できた(実近似率0.9635)。このことはStorage DeviceがIC Memory Card, Semi-Conductor Diskを用いなくてもおおむね記述可能であることを意味している。この差は7件の記事のキー

ワード定義を修正するだけでよい。他にも OS (実近似率 0.9769), Software (0.9804), Network (0.9309), Personal Computer (0.9811) に関して同様の近似が可能であることを確認した。

7.4 包含に基づく近似

包含に基づく近似に関しては、内接率/外接率の数値だけでなく、また内接率と外接率の独立性も考えられる[☆]。しかしこれらは問題領域に依存する可能性が高く、以下では主として前者に注目して2つの近似率(内接率と外接率の組)で実験を行った。1つは(0.90, 1.10), もう1つは(0.80, 1.20)である。

最初の実験では68個の近似を得たのに対して、後の実験では75個の近似結果を得た。以下はこれらの実験の要約である：

状態	対象型数	状態	対象型数
被覆	32	被覆	26
非被覆	5	非被覆	11
欠落	18	欠落	16
再考察	13	再考察	22
合計	68	合計	75
異音同義(被覆)	13	異音同義(被覆)	13
異音同義(他)	14	異音同義(他)	14

本アルゴリズムによって、欠落状態や異音同義の型については新たな ISA 階層の検討事項は生じていない。

これに対して、(実現値の見直しをとまなう)スキーマ修正の候補が、非被覆状態および再考察状態の型から多数得られた。内部近似と外部近似の差がなくなることから候補数が増加したことによるが、(0.90, 1.10)の近似処理で新たに発生した候補は Printing, Component, Information Representation の3つ、(0.80, 1.20)の近似処理で新たに発生した候補は Hyper Text, WordProcessing, Image, Description, Accellarator, Information Management, I/O Control の7つだが、すでに候補であったものでも質的に異なるものが発生している。

たとえば OS は CP/M, DOS/V, MS-DOS, MSX, OS-9, OS/2, OS-Trends, TRON, UNIX, Windows, Windows-NT, ApplicationInterface, NetworkOS, PenComputer, MultiTask, RealTimeOS で完全に近似される(記事数 380)が、(0.90, 1.10)の近似処理では CP/M, DOS/V, MS-DOS, MSX, OS-9, OS/2, OS-Trends, PS/2, TRON, UNIX, WYSIWYG, Windows, ApplicationInterface, NetworkOS, HardDisk, File, PenComputer, MultiTask, Rasterizer,

RealTimeOS, BBS でも近似可能であり(近似率 0.977, 1.071, 記事数 407), この近似処理ではこれらを統合すべきかどうかを問いかけてきた。同じ近似が(0.80, 1.20)の近似処理では CP/M, DOS/V, IC Memory Card, MS-DOS, MSX, OS-9, OS/2, OS-Trends, PS/2, PS/55, TRON, UNIX, WYSIWYG, Windows, ApplicationInterface, Security, NetworkOS, HardDisk, File, PenComputer, Macro, MultiTask, Memory Management, RealTimeOS でも近似可能であり(近似率 0.977, 1.145, 記事数 435), 統合すべき範囲を広げた提案をしてきた。後者では Macro や Security など OS の範囲をどこまで広くとらえるべきかという(ときとして哲学的)問題を提起しているとみられる。これらは共通して変化の激しい分野に相当し、定義の変更を検討すべきであることを示している。

包含に基づく近似アルゴリズムが提示した候補を定義変更と認めれば、該当する記事のキーワードと ISA 階層を修正すればよい。しかし、この例からも分かるように変更を認めるかどうかは問題領域固有の判断であり、近似問題を越えた内容であることに注意したい。

8. 関連研究

スキーマ発見は現在データベース分野で最も大きい研究テーマの1つであり、知識獲得(あるいは機械学習問題)とデータベース設計を結ぶ境界問題である。記述を分析することで新しい型を生成することは意味世界の記述の生成であり、知識発見処理をデータベースに適用した知識獲得技術は、現在多くの研究者の注目を浴びている^{2)~5),9),10),12),30),33)}。たとえば、ヘルシンキ大学 H. Mannila 教授ら¹⁹⁾のグループはこの問題に対して積極的な研究を展開している。

過去データベース研究では、スキーマ進化に関する議論がなされてきたが、これらは主としてデータベースシステムにどのような柔軟な機構を必要とするかというトップダウン的な視点を有している(たとえば文献 7))。筆者の理解する限り、実現値からのフィードバックをとらえ、現在のデータベースを記述するスキーマを得ようとする技術は知られていない。スキーマ発見はメタモデル操作のための一般的な枠組みともとらえられる¹⁴⁾。データベースに対する制約の質問を規則や制約発見と見なせば、これはスキーマ発見の知識獲得そのものである。このきわめて野心的なアイデアは広範囲の応用に適用できるであろう。

データベースにおける知識発見(Knowledge Discovery in Databases)あるいはデータ発見も類似した側面を有するが、データ項目間の相関性や時系列の

[☆] 情報検索分野では「経験的に」相関性が認められているが、上述のように情報検索分野とは性質が異なるため単純な流用はできない。

検出などの統計的成果を求めており、必ずしもスキーマに反映させることを狙ったものではない^{3),27)}。

機械学習 (Machine Learning) の立場からはデータベースを用いた (広義の) 帰納推論が研究されている^{1),28),34)}。これらの大部分はスキーマ発見の手法に通じることが多い。特に属性に基づく概念分類の自動化はよく知られている。たとえば決定木に関しては ID3 や C4.5 などの方法が実用化され、連続値や誤った値 (雑音)、不明値等への対応、ルールベースへの変換など依然として活発な研究が続いている^{31),32)}。決定木の各分岐は概念分類の意図する条件を示しており、スキーマ生成の方法とみることができる。しかしデータベースの立場からはやや奇異なアプローチに思える。たとえば、スキーマ知識、ISA 階層やビュー機能 (第1階述語の論理帰結を表す関連とみられる) をまったく利用せず、データ発生件数に基づいた分類を行う。得られた結果がスキーマに反映されない可能性は高い。帰納的論理プログラムは第1階述語を獲得するための手法であり、属性に基づく概念分類が扱えない問題を論じているが、高速化技術が重要である²⁹⁾。説明による学習⁶⁾は、背景知識として与えられたデータあるいは規則を用い、新たに演繹された規則を補題として活用するという点で帰納的論理プログラムとよく似ている。差別的な汎化による分類基準の確定手法として、版空間モデルが提案されている²⁶⁾。ここではいくつかの条件を組み合わせ、分類階層を説明できる条件を探る効率良い方法を与える。しかしデータを個別に精査して正確に分類することを狙いとするため、雑音や不明値に対応することが難しい。組み合わせるべき基本条件が連続値を含む場合への拡張も自然ではない。

機械学習のうち帰納的推論はスキーマ発見問題と強い関連性を有している^{10),33)}。しかし (制約条件などの) スキーマ情報が主として利用され、若干の例外を除いて実現値の特性や問題領域の知識が利用されない¹⁹⁾。

技術的な観点からは、特性値を用いて型を決定あるいは生成するような暗黙の条件を見いだすことは簡単ではないが、定量的あるいは定性的な方法が知られている³³⁾。因子分析や回帰分析などの統計的な手法は、実現値を分析して一般的な方向を得るのに有用な方法として知られている^{3),13)}。

興味深いアプローチとして、ISA 階層を用いて実現値 (タプル) を解析し新たな述語を見いだす研究がなされている^{12),30)}。ただ属性値の解析を行うものではなく、本研究のように ISA 階層そのものを見直そうという流れにはない。

筆者らの研究は、一貫してスキーマと実現値集合から現状を記述するのにふさわしいスキーマを得るものである。これまで型スキーマ^{21),22)}、述語スキーマ²⁴⁾、複合オブジェクトスキーマ²⁵⁾が統一的なスキーマ発見のパラダイム²³⁾に基づいて展開されていることを明らかにしてきた。過去の知識獲得技法と異なりデータベース的な問題とする根拠は、'大部分のデータベースは効果的であるが、精密にみれば問題が生じており新たなスキーマの発見が差分的に決定できるに違いない'、という視点による。本研究では新たな知識獲得の枠組みを提案したが、依然としてこの観点からデータベース的な問題と考えている。

9. 結 論

本研究では、データベースが実用段階に入ったあとで、型スキーマの発見に関する手法を提案した。得られた結果はデータベース設計段階へのフィードバック情報として重要な役割を果たす。

この目的を達成するため、内部近似および外部近似を組み合わせ、知識 (ISA 階層) に基づく方法、実現値を解析して得る方法および包含状況を吟味する方法を提案した。提案する方法は発見的ではあるが、効果的であることを実験を通じて明らかにすることができた。

これらを直接的に拡張することによって、データベースから第1階述語や複合オブジェクトスキーマを近似する手法を得ることができる。ここで述べた方法は、機械学習手法と似てはいるが、データベーススキーマの進化をスキーマと実現値から行うという点で本質的に異なっている。本手法はこれまでの提案とは異なるまったく新しい知識獲得手法となるもので、データベースとの融合をはかる新たな可能性を有する。

謝辞 本研究に対して激励と貴重なコメントをいただいた増永良文教授 (図書館情報大学) に感謝します。なお、本研究は文部省科学研究費補助金 (重点領域研究高度データベース, 課題番号 09230219) より一部援助をいただいた。

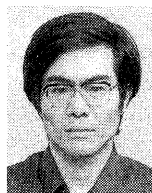
参 考 文 献

- 1) 知識の学習メカニズム, 共立出版 (1986)。
- 2) Agrawal, R., Ghosh, S., et al.: An Interval Classifier for Database Mining Applications, *Very Large DataBase (VLDB)*, pp.560-573 (1992)。
- 3) Agrawal, R. and Srikant, R.: Fast Algorithms for Mining Association Rules, *VLDB*, pp.487-499 (1994)。

- 4) Cai, Y., Cercone, N. and Han, J.: An Attribute-Oriented Approach for Learning Classification Rules from Relational Databases, *Intn'l Conf. on Data Eng. (ICDE)*, pp.281-288 (1990).
- 5) Cai, Y., Cercone, N. and Han, J.: An Attribute-Oriented Induction in Relational Databases, in 33), pp.213-228.
- 6) DeJong, G. and Mooney, R.: Explanation-based Learning - An Alternative View, *Machine Learning 1*, pp.145-176 (1986).
- 7) Elmasri, R. and Navathe, S.B.: *Fundamentals of Database Systems*, Benjamin (1989).
- 8) Embley, D. et al.: *Object Oriented System Analysis*, Yourdon Press (1992).
- 9) Frawley, W.J., Piatetsky-Shapiro, G. and Matheus, C.J.: Knowledge Discovery in Databases - An Overview, in Ref. 33), pp.1-30.
- 10) Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R. (Eds): *Advances in Knowledge Discovery and Data Mining*, MIT Press (1996).
- 11) Hull, R. and King, R.: Semantic Data Modeling, *ACM Computing Surveys*, 19-3, pp.201-260, ACM (1987).
- 12) Han, J., Cai, Y. and Cercone, N.: Knowledge Discovery in Databases: An Attribute Oriented Approach, *VLDB*, pp.547-559 (1992).
- 13) Han, J. and Fu, Y.: Discovery of Multiple-Level Association Rules from Large Databases, *VLDB*, pp.420-431 (1995).
- 14) Imielinski, T. and Mannila, H.: A Database Perspective on Knowledge Discovery, *CACM* 39-11, pp.58-64, ACM (1996).
- 15) 石塚：知識の表現と高速推論，丸善出版 (1995)。
- 16) Kim, W.: *Introduction to Object Oriented Databases*, MIT Press (1990).
- 17) Korth, H.F. and Silberschatz, A.: *Database System Concepts*, McGraw-Hill (1986).
- 18) Langley, P.: Machine Learning and Concept Formation, *Machine Learning*, Vol.3, pp.99-102 (1989).
- 19) Mannila, H.: Methods and Problems in Data Mining, *Intn'l Conf. on Database Theory (ICDT)*, pp.41-55 (1997).
- 20) Miura, T.: Database Paradigms Towards Model Building, *Object Roll Modelling*, pp.228-258 (1994).
- 21) 三浦，塩谷：データベースにおける型スキーマの発見，情報処理学会論文誌，Vol.38, No.6, pp.1172-1181 (1997).
- 22) Miura, T. and Shioya, I.: Knowledge Acquisition for Classification Systems, *Tools with Artificial Intelligence (ICTAI)*, pp.110-115 (1996).
- 23) Miura, T. and Shioya, I.: Paradigm for Schema Discovery, *Intn'l Symposium on Cooperative Database Systems for Advanced Applications (CODAS)*, pp.101-108 (1996).
- 24) Miura, T. and Shioya, I.: Differentiation for Schema Discovery, *Intn'l Database Workshop*, pp.62-77 (1997).
- 25) Miura, T. and Shioya, I.: Examining Complex Objects for Type Schema Discovery, *Conference and Workshop of DEXA*, pp.462-477 (1997).
- 26) Mitchell, T.: Version Space - A Candidate Elimination Approach to Rule Learning, *IJCAI* (1977).
- 27) 森下：データマイニングと最適化技術について，信学技報，AI97-50, DE97-83 (1997)。
- 28) 元田，鷲尾：機械学習とデータマイニング，人工知能学会誌，Vol.12, No.4, pp.505-512 (1997)。
- 29) Muggleton, S.: Inductive Logic Programming, *Inductive Logic Programming*, Muggleton, S. (Ed.), pp.3-27, Academic Press (1992).
- 30) Ng, R. and Han, J.: Efficient and Effective Clustering Methods for Spatial Data Mining, *VLDB*, pp.144-155 (1994).
- 31) Quinlan, R.: Induction of Decision Trees, *Machine Learning*, Vol.1, No.1, pp.81-106 (1986).
- 32) Quinlan, R.: Learning Logical Definition from Rules, *Machine Learning*, Vol.5, No.3, pp.239-266 (1990).
- 33) Piatetsky-Shapiro, G. and Frawley, W.J. (Eds): *Knowledge Discovery in Databases*, MIT Press (1991).
- 34) Wu, X.: *Knowledge Acquisition from Databases*, Ablex Publishing (1995).

(平成 9 年 8 月 20 日受付)

(平成 10 年 2 月 2 日採録)



三浦 孝夫 (正会員)

京都大学理学部，工学博士（東京大学）。1998 年 4 月より法政大学工学部電気電子工学科教授。データモデル，知識表現，演繹データベース，複合オブジェクトなどの分野の研究

に従事。電子情報通信学会，ACM 各会員。



塩谷 勇 (正会員)

東京電機大学大学院修士課程修了。現在，産能大学経営情報学部助教授。論理プログラミング，グラフ文法，論理データベースの研究に従事。電子情報通信学会，ACM 各会員。