論　文　　Technical Papers

# 完全・不完全知識を扱う高水準論理型データベースの自己組織化

## Self Organization of Advanced Logical Database Handling Complete-Incomplete Knowledge

井戸　譲治*　馬場口　登*　北橋　忠宏*
George Ido　Noboru Babaguchi　Tadahiro Kitahashi

* 大阪大学産業科学研究所
I. S. I. R., Osaka University, Ibaraki, Osaka 567, Japan.

1994年4月26日　受理

**Keywords:** self organization, incomplete knowledge, logical database, nonmonotonic reasoning, consistency.

**Summary**

In this paper, we discuss a self organization mechanism of ALDB (Advanced Logical DataBase) that is capable of handling the complete-incomplete knowledge. A number of exceptional data will be included as a DB gets larger. The knowledge including exceptions can be considered as the incomplete knowledge, because it may derive inconsistency. For logical DB, the occurence of inconsistency will be crucial. Therefore, in order to maintain the consistency of DB, some self organization mechanism should be developed. We propose the self organization mechanism (SOM) of ALDB which is realized by transforming the complete knowledge into the incomplete knowledge.

The ALDB consists of extensional DB (EDB), complete intensional DB (CIDB) and incomplete intensional DB (IIDB). EDB is a set of facts, whereas CIDB and IIDB are sets of rules representing complete knowledge and incomplete knowledge, respectively. The data retrieved from the ALDB is concerned with the ALDB extension that is a set of derived conclusions.

The SOM is divided into the three steps: 1) the check of consistency, 2) the selection of a rule to be transformed and 3) the transformation to an incomplete rule and its ordering. When new data is added into the ALDB, the SOM firstly checks the consistency by investigating only the part of complete knowledge with the use of SLD resolution technique. If the ALDB is inconsistent, the SOM identifies a complete rule that is logically false by contradiction backtracking algorithm. The selected rule is then transformed to an incomplete rule whose intended meaning is "If…, then normally…". It allows the exceptions that are not satisfied with a complete rule. Finally, the SOM makes the ordering among the incomplete rules based on the generality of each rule. The ordering plays a significant role to reduce the number of ALDB extensions.

## 1. はじめに

近年，高水準の問合せ処理を実現し得るデータベースとして，演繹データベース(DDB: Deductive Database)[Minker 87]が注目を集めている．演繹DBは形式論理に基礎を置く論理型データベース(LDB: Logical DB)の代表的モデルであり，事実の集合である外延データベース(EDB: Extensional DB)とルールの集合である内包データベース(IDB: Intensional DB)，およびEDB，IDBの操作モジュールから構成される．DB検索に関連するさまざまな処理が一階述語論理の枠組みで特徴づけられるので，演繹DBは，データベース理論における有力な数学的モデルとなる．

ここで，演繹DBを知識システム的に捉えると，演繹DBは通常論理に基づく演繹推論エンジンを用いる

システムといえる．演繹推論は本質的に，常に正しい完全な知識を対象とする推論機構であり，モデリング世界に関してすべての知識の記述が不可欠となる．現実世界の多様なデータに関する知識は，一般に不完全であるとの認識に立つと，演繹DBに対する限界が示唆される．

一方，DBの大規模化に伴って，例外データやノイズの混入は不可避となる．ここで，例外について考えよう．例外とは，それ自身誤りではないものの，ルールに従わないインスタンスのことを指し，例外の存在はDBでの矛盾を誘発する要因となる．知識ベースに例外を含むとき，例外に関係するルールは常に正しいとは限らないことから，例外を含む知識は不完全な知識の一例といえる．

さて，一階述語論理などの通常論理において，知識集合に矛盾を含むときは，すべての論理式が導き出されるので，演繹DBでの矛盾の発生は致命的な障害となる．したがって，DB内の矛盾を検出し，適切な修正を行って無矛盾性を回復する機能が不可欠となる．

DBの規模が小さい場合，矛盾の原因となる例外データを逐一排除するという便宜的方策も利用できよう．しかし，DBが大規模な場合には，矛盾を回避する目的で，ルールやデータを無制限に排除する方法では，知識ベースの漸増的な充実はとうてい期待できない．ゆえに，例外を適切にハンドリングしながら知識ベースを順次拡充していく非単調知識展開技法を実現することが望まれている[Esculier 90, 西尾 93]．

以上の背景から，筆者らはDBシステムのいっそうの高機能化に向けて，完全な知識と不完全な知識とが共存する内包DB，および非単調性を有する推論エンジンを装備した高水準論理型データベースALDB (Advanced LDB)を提案し，表現形態・問合せ手続きなどの基本設計を行ってきた[馬場口 94, 井戸 93]．本論文では，前述の非単調知識展開実現への一アプローチとして，ALDBにおける自己組織化手法を議論する．提案手法は，新規追加データに対して矛盾発生状況を管理し，矛盾が発生したとき，その要因となる完全な知識を不完全な知識へとルール変換[井戸 94]することにより無矛盾性を回復するものである．

このルール(知識)変換の問題は，人間が完全な知識や不完全な知識をどのように獲得していくかという認知科学的プロセスとの関連からも興味深い問題である．すなわち，不完全知識が完全知識に変化するのか，その逆か，あるいは両方が融合しているのかなど，議論の余地が多い．本論文では，完全から不完全に移行する立場から考察していく．

## 2. ALDBの概要

### 2・1 ALDBの知識表現

ALDBは，論理型DBの一モデルであるので演繹DBと同様に，ファクト集合であるEDBとルール集合であるIDBから構成される．さらに，ALDBではIDBが，完全知識を表現する完全IDB (CIDB: Complete IDB)と不完全知識を表現する不完全IDB (IIDB: Incomplete IDB)の二つに分けられる．

ここでは，知識表現をルール形式に統一するという設計方針を掲げ，完全な知識にはDDBのルール形式(完全ルールと呼ぶ)を踏襲し，不完全な知識には，例外許容型のオペレータを伴うデフォルトルール形式(不完全ルールと呼ぶ)を採用する．以下に，本ALDBにおけるEDBとIDBの表現形式を示す．

**【定義1】 ALDBの表現形式** ALDBのEDB, CIDB, IIDBはおのおの，以下のファクト，完全ルール・制約，不完全ルールの集合である．

(i) ファクト：$p(t_1, \cdots, t_n)$ ($p$は$n$項述語記号，$t_1, \cdots, t_n$は定数)

(ii) 完全ルール：$A \leftarrow B_1, \cdots, B_m$ ($A, B_1, \cdots, B_m$は原子式)

(iii) 制約：$\perp \leftarrow C_1, \cdots, C_m$ ($C_1, \cdots, C_m$は原子式，$\perp$は矛盾を表す特殊な述語)

(iv) 不完全ルール：$A \Leftarrow B_1, \cdots, B_m$ ($A, B_1, \cdots, B_m$は原子式)

ただし，各ルールには以下の制限を加える．完全・不完全ルールは関数を含まず，ルールの頭部$A$に現れる変数はすべて，ルールの本体$B_1, \cdots, B_m$に現れ，変数($X, Y, \cdots$などで表す)は全称限量されているとする．□

完全ルール$A \leftarrow B_1, \cdots, B_m$における記号$\leftarrow$は通常論理での含意記号である．よって，そのルールは，「$B_1, \cdots, B_m$ならば**必ず**$A$である」を表す．上の定義よりわかるように，ALDBのルールはすべて原子式から構成される，いわゆるホーン節形式である．制約なる特殊な完全ルールを導入する理由は，このような制限のもとでは否定情報が導出されないからである．制約は「$C_1, \cdots, C_m$が**同時に成り立つ**ことは**矛盾**する」を意味するもので，演繹DBにおける整合性制約と同様の働きをする．

不完全ルール$A \Leftarrow B_1, \cdots, B_m$の意味は，「$B_1, \cdots, B_m$ならば**通常**$A$である」とする．不完全ルールの意味は自己認識論理(Autoepistemic Logic)[Moore 85]に従う．自己認識論理の言語体系は，「信じている

(believed)」という意の様相記号Lを含めて定義されているが，ここで利用する不完全ルールにはLの代わりに，例外許容型含意記号⇐を用いる．

**【定義2】** ALDBにおける不完全ルール $A \Leftarrow B_1, \cdots, B_m$ と自己認識論理式 $A \leftarrow LB_1, \cdots, LB_m, \neg L \neg A$ とは等価である．ここで，$LB_i$ は，$B_i$ がDB(論理式集合)から導かれるとき真，$\neg L \neg A$ は，$\neg A$ がDBから導かれないとき真と解釈する． □

例外を含む知識，例えば「鳥は通常飛ぶ」を不完全ルールで表すと，

$$\mathrm{fly}(X) \Leftarrow \mathrm{bird}(X).$$

になる．この不完全ルールに，

$$\mathrm{bird}(X) \leftarrow \mathrm{penguin}(X)$$
$$\mathrm{not\text{-}fly}(X) \leftarrow \mathrm{penguin}(X)$$
$$\bot \leftarrow \mathrm{fly}(X), \mathrm{not\text{-}fly}(X)$$
$$\mathrm{penguin}(\mathit{tweety}), \mathrm{bird}(\mathit{sweety})$$

という論理式集合を加えても，矛盾しない．すなわち，"飛ぶ"という点について，ペンギンは鳥の例外であるが，不完全ルールは例外を許容しつつ，一般的な知識も表現できる．仮に「鳥は飛ぶ」を完全ルール $\mathrm{fly}(X) \leftarrow \mathrm{bird}(X)$ で記述すると，矛盾が生じてしまう．

ALDBは自己認識論理の部分体系であり，自己認識論理における解釈，モデル，拡張などの概念は，ALDBにおいてもそのまま適用される．言い換えれば，ALDBは自己認識論理を基礎とする論理型DBである．

### 2・2 ALDB拡張と問合せ

ALDBに含まれる知識(EDB, IDB)から推論される結論集合であるALDB拡張を次のように定義する．

**【定義3】 ALDB拡張** 以下の条件を満たす論理式集合 $T$ をALDB拡張という．

$$T = Th[\mathrm{EDB} \cup \mathrm{CIDB} \cup Con(\mathrm{IIDB})]$$

ただし，$Con(\mathrm{IIDB}) = \{A \mid A \Leftarrow B_1, \cdots, B_n \in \mathrm{IIDB}, B_1, \cdots, B_n \in T, \neg A \notin T\}$．また，$Th$ は演繹的閉包を表す記号である． □

ALDB拡張は演繹DBにおける定理(theorem)集合と同等の概念である．なお，ALDB拡張は自己認識論理の拡張(extension)をALDBにマップさせたもので，任意のALDBは，一つ以上のALDB拡張を持つという好ましい性質*1がある[井戸 94]．

問合せに対するALDBの応答は，ALDB拡張と問合せを表す論理式とのメンバシップ関係に依存する．基礎リテラルの問合せに対しては，ALDB拡張に含まれるか否かに従い，"Yes", "No"を返答する．変数を含むリテラルの問合せに対しては，変数にインスタンスを代入して返答する．

筆者らはすでに，反駁証明プロセスを利用した問合せ手続きを提案している．その手続きは基本的に，ある式のALDB拡張に対するメンバシップ関係を，その式のSLD導出に基づく反駁の有無により決定するものである．この反駁生成の過程では，不完全ルール $A \Leftarrow B_1, \cdots, B_m$ を自己認識論理式 $A \leftarrow LB_1, \cdots, LB_m, \neg L \neg A$ として扱っており，サブゴール $LB_i, \neg L \neg A$ に対しては，それぞれ $B_i, \neg A$ に対するSLD導出木を副次的に生成する．ここで，負リテラルの証明が必要となるが，通常のSLD導出では正リテラルの連言の証明しか行えないので，制約を変形して負リテラルを頭部に持つルールを作成し，導出を実行する．手続きの詳細については[井戸 93]を参照されたい．

### 2・3 ALDBの無矛盾性

ALDBの無矛盾性を以下のように定義する．

**【定義4】 ALDBの無矛盾性** ALDB拡張がそれぞれ無矛盾であるとき，またそのときに限り，ALDBは無矛盾であるという． □

このように，ALDBの無矛盾性を，推論結果であるALDB拡張を通して定義する．また，ALDB拡張の無矛盾性に関して重要な定理を示す．

[**定理1**] ALDBの完全知識部(ファクト，完全ルール)の集合，すなわちEDB∪CIDBが矛盾するとき，またそのときに限り，ALDB拡張が矛盾する． □

《**証明**》 第一に，EDB∪CIDBが矛盾するとする．このとき，$Th$[EDB∪CIDB]はすべての論理式を含む．ALDB拡張に関する定義3より $Th$[EDB∪CIDB]はALDB拡張の部分集合となるので，ALDB拡張はすべての論理式を含む．したがって，ALDB拡張は矛盾する．

第二に，ALDB拡張を表す論理式集合を $T$ として，$T$ が矛盾するとする．$T$ は演繹的閉包のもとに閉じているから，すべての論理式を含む．したがって，$\neg A \notin T$ なる不完全ルール $A \Leftarrow B_1, \cdots, B_m$ は存在せず，$Con$(IIDB)は空となる．よって $T = Th$[EDB∪CIDB]．ここでEDB∪CIDBが無矛盾であると仮定すると，$p \in Th$[EDB∪CIDB]かつ $\neg p \notin Th$[EDB∪CIDB]なる論理式 $p$ が存在する．すると $p \in T$ かつ $\neg p \notin T$ となり，$T$ がすべての論理式を含むことに矛盾する．よって，EDB∪CIDBが矛盾することが示された． □

本定理により，拡張の無矛盾性を，実際に拡張を構成することなく，しかも完全知識部のみを対象として

*1 自己認識論理の拡張は存在しないことがある．ただし，拡張を複数持つ場合(多重拡張)があることは，自己認識論理でもALDBでも同じである．

検証できることが保証される．

## 3．ALDBの自己組織化

ALDBが不完全な知識を陽に扱うことの効用は，例外が混入してもルールを廃棄することなくDBの無矛盾性を維持し，なおかつ例外のみを排除した巧みな問合せを実現し得ることにある．例外により生じる矛盾への対処方法を考えたとき，あらゆる例外を考慮して最初からルールすべてを不完全ルールにしておくことが想定される．しかしながら，不完全ルールを用いる推論に要する計算コストは，完全ルールのそれに比べるとかなり高いうえ，不完全ルールの増加はALDB拡張の数の増加につながるため，不完全ルールは無矛盾性維持のために必要最小限だけ保持するのが妥当と考えられる．ただ，どのルールが例外を含むか，すなわちどのルールが不完全であるかを初期段階で的確に判断することは不可能に近い．

そこで，本論文では，完全ルールが既存のものであるとし，矛盾が生じたときに完全ルールを変化させ不完全化するというアプローチをとる．ALDBの漸進的な拡充に関して，以下のような仮定を置く．

- ALDBに新規追加されるデータは，ファクト，完全ルール，制約のみとする．
- 不完全ルールは，DBの無矛盾性を維持するためにALDB自身が内部で完全ルールをもとに生成するので，外部から不完全ルールが入力されることはない．

このように仮定することで，不完全ルールの生成を最小限にして，処理効率の低下，多重拡張の発生をある程度抑制することを図る．以上のように，新規データに伴う矛盾を発見し，データベースを修正して無矛盾性を回復する機能を，ここでは自己組織化(self organization)と呼ぶ[井戸 94]．

図1にALDBの自己組織化のフローを示す．ファクト，完全ルールがALDBに追加された後，1)無矛盾性検査，2)変換対象ルールの選択，3)ルール変換を経て，再び無矛盾な状態に復帰する．以下，これらについて詳述する．

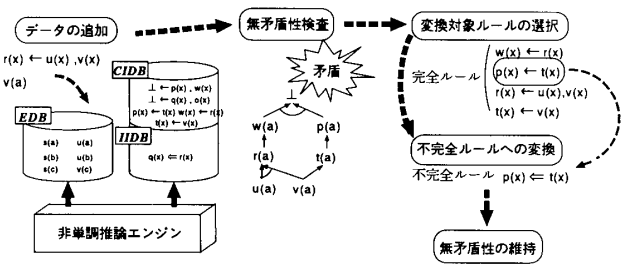


図 1　ALDBの自己組織化

### 3・1　無矛盾性検査

無矛盾性検査は，新規追加データによりALDBが無矛盾か否かを検査し，もし矛盾していれば，矛盾を引き起こす原因となるルールの部分集合を求める処理である．

定義4より，ALDBの無矛盾性は，ALDBから導かれる結論の集合であるALDB拡張の無矛盾性を通して定義されている．ALDB拡張の無矛盾性に関して，拡張を構成して直接調べることなく，しかも完全知識部のみを対象として検証できることが，定理1により保証される．したがって，無矛盾性の判定に，導出などの一階述語論理の定理証明技法が活用できる．また，定義1で示した制限は，ALDB表現に対する導出の停止性を保証するためのものである．

特に，ALDBではホーン節集合を対象としているので，SLD導出を用いることができる．ホーン節集合では，矛盾する式集合には唯一の制約が含まれるので，制約を頂上節とするSLD導出を行い，これが空節に至れば，そのとき用いられた側節と制約が矛盾していることになる．

制約を頂上節とするSLD導出により無矛盾性を判定する場合，新規に追加されたデータがどの制約に関係するかは，別に情報を持たせない限り不明である．そのような場合は原則的に，すべての制約について導出を行わなければならない．これは効率面で得策ではないので，無関係な制約を処理の対象から除外する工夫を施す．

ここでは，各述語にそれ自身が影響を及ぼす制約を登録しておくようにする．すなわち，各述語に対し，

〈述語，{制約リスト}〉

の形式で記録する．これを実現するには，制約から始めて，ルール本体に現れる述語を頭部に持つファクト，ルールを順次呼び出し，呼び出されたファクト，およびルールの頭部の述語が呼び出した制約を登録すればよい．例えば，

(1)　$\perp \leftarrow p(X, Y), q(X), r(Y)$
(2)　$\perp \leftarrow s(X), t(X)$
(3)　$\perp \leftarrow u(X, Y), v(X, Y)$
(4)　$p(X, Y) \leftarrow pa(X), pb(Y)$
(5)　$q(X) \leftarrow qa(X)$
(6)　$q(X) \leftarrow qb(X)$
(7)　$s(X) \leftarrow pa(X)$
(8)　$t(X) \leftarrow v(X, Y), qb(Y)$

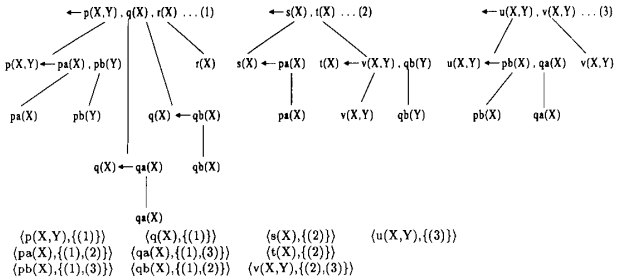


**図 2** 述語への制約の登録

(9) $u(X, Y) \leftarrow pb(X), qa(X)$

という場合，述語への制約の登録は図2のようになる．提案の登録法は，演繹DBでよく使われる依存グラフにおいて，始点が注目する述語で終点が述語⊥の連結パス(依存関係のあるパス)に着目して，始点の述語と終点に関与する制約を記述していることに等しい．

新規データとして例えば $p(X, Y) \leftarrow pc(X), pd(Y)$ が追加される場合，三つの制約のうち $p(X, Y)$ が影響するのは(1)だけであるので，(1)についてのみ検査すればよい．最終的にこのデータがALDBに組み込まれる場合，その本体の述語 $pc(X), pd(X)$ も制約(1)に影響を及ぼすため，$\langle pc(X), \{(1)\}\rangle, \langle pd(X), \{(1)\}\rangle$ とする．

### 3・2 変換対象ルールの選択

抽出された完全知識部のルールの部分集合から，不完全ルールに変換すべき完全ルール(変換対象ルール)を選び出す処理について記述する．

無矛盾性検査は一階ホーン節集合で表される完全知識部に対して適用されるため，矛盾が生じるときには必ずSLD反駁木が生成される．本論文では，オラクル(基礎原子式の真偽に関する返答を行う機構)を仮定し，Shapiroの矛盾点追跡(contradiction backtracking)アルゴリズム[Shapiro 81]を応用して変換対象ルールの選択を実行する．ここで，オラクルはユーザとの質問応答により実現するものとする．

矛盾点追跡アルゴリズムは，SLDの反駁木の根に位置する空節から出発して，意図された解釈(intended interpretation)のもとでの基礎原子式の真偽を与えながら，反駁木を遡行して論理的に誤ったルールを特定するものである．SLD反駁木は，その内部節点に対する子のうち，少なくとも一つが葉である特殊な2分木である．反駁木の中心節はゴール節(制約)からなり，側節はファクトないしは完全ルールである．アルゴリズムでは，導出に用いた原子式の変数に基礎代入を行うことによって，得られた基礎原子式に対して，その原子式がEDBに含まれるときには *true*[*2]，そうでないときにはオラクルに基づき，その *true*, *false* を決定する．定められた原子式の真偽に従い，*true* のとき中心節へ，*false* のとき側節へさかのぼる．このようにして葉に到達したとき，その葉が表すルール(制約は除く)が選択されるルールである．ここで，ルールの真偽を直接ユーザに質問するのではなく，個々の基礎原子式の真偽を質問することに注意されたい．

### 3・3 ルールの変換

矛盾点追跡アルゴリズムにより選択された完全ルールを不完全ルールに変換することにより，無矛盾性を回復する．ここで完全ルールから不完全ルールへの変換に関して次の定理を与える．

[**定理2**] $S'$=EDB∪CIDBが無矛盾，$S' \cup \{A \leftarrow B_1, \cdots, B_m\}$ が矛盾するとする．このとき，$S' \cup \{A \Leftarrow B_1, \cdots, B_m\}$ のALDB拡張は無矛盾である． □

《**証明**》 定理1からALDBの無矛盾性は完全知識部の無矛盾性に依存する．いま，$S'$ が無矛盾であるから明らかにALDB拡張は無矛盾である． □

これは演繹DBで，あるルールを破棄することにより矛盾が解消する場合に対応して，ALDBではそのルールを不完全ルールに変換することで，ルールを破棄することなく矛盾が解消できることを示している．

不完全ルールが新たに生成されたときには，既存の不完全ルールとの順序づけを行う．前述のように，ALDBにおいては拡張が複数存在することがある．多重拡張は推論結果として不適切なものを含んでしまうため，そのような場合にはより適切な拡張のみを選びださなければならない．そのため，多重拡張の原因となる不完全ルール間に順序関係を設定し，これに基づいて拡張を選び出すようにする．

**【定義5】 不完全ルールの順序づけ** 不完全ルール $R$ の本体を $Body(R)$ と表すことにする．不完全ルール $R_1, R_2$ に対し，

$$Body(R_2) \subset Th[\text{EDB} \cup \text{CIDB} \cup Body(R_1)]$$

かつ

$$Body(R_1) \not\subset Th[\text{EDB} \cup \text{CIDB} \cup Body(R_2)]$$

が成り立つとき，$R_1$ のほうが $R_2$ より優先するといい，その順序づけを $R_1 < R_2$ と書く． □

この定義は，二つの不完全ルールが存在するとき，おのおののルールの本体の一般性を比較して，特殊なものほど優先度が高いことを意味する．

以上の順序づけによって，$R_1 < R_2$ なる不完全ルール $R_1, R_2$ に対して，それぞれ一方を適用することにより

*2 本論文では，ファクトにノイズは含まれないものと仮定する．

二つの拡張が得られる場合は，$R_1$ を適用して得られる拡張を優先する．

## 4. 動作例と検討

**前章**までで述べた自己組織化の例を示し検討を加える．

### 4・1 動　作　例

EDB : swallow(*marcus*), swallow(*mike*),
dog(*pat*)
(*marcus, mike, pat* は定数を表す)

CIDB : fly($X$)← bird($X$)
bird($X$)← swallow($X$)
animal($X$)← bird($X$)
animal($X$)← dog($X$)
⊥← fly($X$), not-fly($X$)

IIDB : not-fly($X$)⇐animal($X$)

を考えると，このALDBからは

swallow(*marcus*), bird(*marcus*), animal(*marcus*), fly(*marcus*), swallow(*mike*), bird(*mike*), animal(*mike*), fly(*mike*), dog(*pat*), animal(*pat*), not-fly(*pat*)

が導かれる．以上の導出データのうち，swallow(*marcus*), swallow(*mike*), dog(*pat*)がEDBのファクトであり，not-fly(*pat*)が不完全ルールにより導かれたもの，そして残りの原子式は完全ルールによって導かれたものである．

ここで新規データとして

not-fly($X$)← bird($X$), injured($X$)
injured(*mike*)

が追加されるとする．

第一に，無矛盾性検査を行うが，すでに述べたように，不完全ルール not-fly($X$)⇐animal($X$)を考慮する必要はない．制約⊥← fly($X$), not-fly($X$)からSLD導出を行うと図3のような反駁木(図中，空節は□で表しているが意味は⊥と同じ)が成立し，矛盾していることがわかる．

第二に，この反駁木に対し，矛盾点追跡アルゴリズムを適用する．まず，injured(*mike*)の真偽が問われるが，これはEDBファクトなので *true* となり，中心節のほうにさかのぼる．次にbird(*mike*)の真偽はオラクルによって *true* と与えられ，同様に中心節をさかのぼる．このように，not-fly(*mike*)を *true*(オラクル)，swallow(*mike*)を *true*(ファクト)，bird(*mike*)を *true*(オラクル)，fly(*mike*)を *false*(オラクル)として

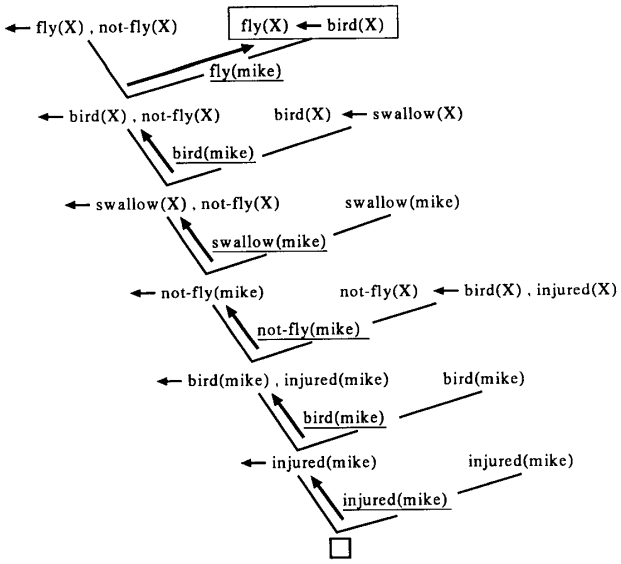


**図 3**　SLD反駁木と矛盾点追跡

アルゴリズムを実行すると，

fly($X$)← bird($X$)

が選択される．図3に矛盾点追跡のようすを示すが，同図中，下線部が真偽を問われる基礎原子式で，太い矢印がその真理値に従ってさかのぼる道筋を表す．

第三に，選択されたルールを不完全ルール

fly($X$)⇐bird($X$)

に変換し既存の不完全ルールnot-fly($X$)⇐animal($X$)との間に順序づけを行うと，EDB∪CIDB∪bird($X$)からanimal($X$)は導かれるが，EDB∪CIDB∪animal($X$)からはbird($X$)は導かれないので

fly($X$)⇐bird($X$)＜not-fly($X$)⇐animal($X$)

となる．'bird'と'animal'を比較すると概念上は'animal'のほうが上位にあるので，このような順序づけがなされるのである．したがって，自己組織化後のALDBは次のようになる．

EDB : swallow(*marcus*), swallow(*mike*),
dog(*pat*), injured(*mike*)

CIDB : fly($X$)← bird($X$)
bird($X$)← swallow($X$)
animal($X$)← bird($X$)
animal($X$)← dog($X$)
not-fly($X$)← injured($X$)
⊥← fly($X$), not-fly($X$)

IIDB : not-fly($X$)⇐animal($X$)
fly($X$)⇐bird($X$)

このときALDB拡張は，fly($X$)⇐bird($X$)を適用した場合に得られるfly(*marcus*)を含むものとnot-fly($X$)⇐animal($X$)を適用した場合に得られるnot-fly(*marcus*)を含むものの二つが存在するが，不完全ルールの順序づけにより，前者が優先される．また，これらはいず

れも

swallow(*marcus*), bird(*marcus*), animal(*pat*), swallow(*mike*), bird(*mike*), animal(*mike*), not-fly(*mike*), injured(*mike*), dog(*pat*), animal(*marcus*), not-fly(*pat*)

を含んでおり，無矛盾性が回復され，しかも矛盾に無関係なインスタンスは影響を受けていないことがわかる．

### 4・2 検　　討

#### 1) ルール変換について

論理型DBで矛盾が生じたとき，これを解消するには広い意味での特殊化(DBから導かれるデータ集合を小さくする)が必要となる．例えば，ルール fly($X$)← bird($X$)が論理的に誤ったルールとして判明したとき，最も単純な特殊化はこれを削除することである．このとき矛盾の原因となるインスタンスが導かれなくなるため矛盾は解消するが(前節の例では fly(*mike*))，矛盾に無関係なインスタンスに関する知識(前節の例では fly(*marcus*))まで導かれなくなるという問題点がある．

次に想定されるのは，例外事例を明示的にルールに挿入する手段である．例えば，飛ぶことに関する鳥の例外であるペンギンを除くことを意図して，一般的ルールを fly($X$)← bird($X$),¬ penguin($X$)に修正すると，先と同様に導かれないデータが存在する．この手段もやはり過特殊化となる．

これらは一階述語に制限しているために生ずる問題であり，このような例外的な事例を扱いつつ適切な特殊化を行うには，例外についての膨大な記述(ペンギンでなく，しかもひなでなく，しかも翼が傷ついておらず…)の付加を余儀なくされる．また，例外でないことを明示的にルールに付加するため，何が例外かを常に把握しておかなければならない．これに対して，提案の不完全ルールに変換する手法は特に，あらかじめ例外とは認識されていない例外的データにも，無矛盾性の維持という側面から，適正に対処可能である．しかしながら，不完全ルールの導入についての計算量の増大は無視できず，効率的な制御ストラテジーを設定することが必要である．

#### 2) 矛盾点追跡アルゴリズムの拡張

矛盾点追跡アルゴリズムを用いる際，ユーザに基礎原子式の真偽を質問しているが，あらゆる基礎原子式の真偽をユーザが把握していると仮定するのは，強すぎる制限と考えられる．ここでは，基礎原子式の真偽に関して，*true*(真)，*false*(偽)に加え，*unknown*(不明)を扱えるようアルゴリズムを拡張する．

矛盾点追跡アルゴリズムでは，導出に用いた基礎原子式の真偽により，中心節をさかのぼるか側節をさかのぼるかが決定される．*unknown* の場合には，両方の枝をさかのぼればよい．すると複数の変換対象ルールが選択されることになるが，これらは結論部が互いに矛盾するため，結果的に多重拡張となる．また，各拡張は *unknown* とされた基礎原子式を個々に含む．

次のような矛盾するALDBを考えよう．

EDB：quaker(*nixon*), republican(*nixon*)

CIDB：pacifist($X$)← quaker($X$)

not-pacifist($X$)← republican($X$)

⊥← pacifist($X$), not-pacifist($X$)

図4のようなSLD反駁木において，republican(*nixon*)および quaker(*nixon*)を *true*, not-pacifist(*nixon*)および pacifist(*nixon*)を *unknown* とする．空節から出発して，まず republican(*nixon*)は *true* なので中心節へ進み，次の not-pacifist(*nixon*)は *unknown* なので両方の枝に進む．まず側節のほうは葉であり，not-pacifist($X$)← republican($X$)が変換の候補となる．中心節の方向へさらに矛盾点追跡を続け，quaker(*nixon*)は *true* なので中心節に進み，次の pacifist(*nixon*)は *unknown* なので両方の枝をさかのぼる．ここで，どちらも葉に達するので，⊥← pacifist($X$), not-pacifist($X$)と pacifist($X$)← quaker($X$)がともに変換の候補となる．ただし，制約は不完全ルールに変換できないので，ここでは二つの完全ルールが不完全ルールに変換され，更新後の ALDB は以下のようになる．

EDB：quaker(*nixon*), republican(*nixon*)

CIDB：⊥← pacifist($X$), not-pacifist($X$)

IIDB：pacifist($X$)⇐quaker($X$)

not-pacifist($X$)⇐republican($X$)

ALDB拡張として quaker(*nixon*), republican(*nixon*), pacifist(*nixon*)を含むものと，quaker(*nixon*), republican(*nixon*), not-pacifist(*nixon*)を含むものの二つが

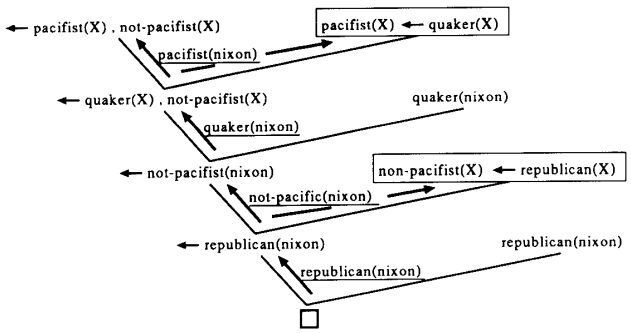


**図 4**　unknown を用いた矛盾点追跡

存在する．この場合は，不完全ルール間の順序づけができない場合である．拡張を少なくすること，理想的には一つにすることが望ましいと思われるが，ルールの順序づけのみでは対応できないことを示唆している．例えば，層状DB[Minker 87]のように，述語間の順序づけも解決策として想定されるが，それらの全順序を定めることは現実的に極めて難しいであろう．したがって，ユーザの選好をインタラクティブに反映させるなど，実用的な対策を検討しなければならない．

## 5. むすび

本論文では，完全な知識と不完全な知識の双方を操作できるALDBの自己組織化機構を提案した．提案手法は，論理型DBの中心的課題である矛盾への対処法を示したもので，例外を含む知識を表現できる不完全ルールの特徴を利用して，完全ルールから不完全ルールへの変換により無矛盾性の維持を図るものである．提案手法は，ワークステーション上にPrologにより実装されており，良好に動作することを確認している．このような動的にルールを改定するメカニズムは，知識データベース構築方法論を議論するうえできわめて重要な部分を占めると考えている．

さて，人間においては，正しいと思っていた知識が，現実の問題解決の場面で食違いを生じたことから，知識の内容を変更させることがある．提案手法は，このようなプロセスを説明する一つの枠組みといえよう．また，不完全知識や非単調推論に関する従来研究では，「まず不完全知識ありき」で「不完全な知識を用いた推論をどう定式化するか」が主な関心であった．「不完全な知識をどう生成するか」は，ほとんど議論されずにきた．筆者らは本研究を不完全知識生成の一試案と位置づけている．

今後の課題として，多重拡張の扱い，実データベースでの評価などがあげられる．最後に，本研究の一部は，文部省科学研究費，電気通信普及財団の補助によることを付記する．

## ◇参考文献◇


[馬場口 94] 馬場口登，大川剛直：完全/不完全知識を扱う高次推論型データベースにおける知識獲得，文部省科学研究費重点領域(知識科学)成果報告論文集，pp. 392-399 (1994).

[Esculier 90] Esculier, C.: Non-monotonic Knowledge Evolution in VLKDBs, *Proc. 16th VLDB Conf.*, pp. 638-649 (1990).

[井戸 93] 井戸譲治，馬場口登：完全/不完全なルールに対する推論手続き，情処学研報，93-AI-86-3, pp. 17-24 (1993).

[井戸 94] 井戸譲治，馬場口登，北橋忠宏：完全知識から不完全知識へのルール変換について，信学研報，AI-93-73, pp. 9-16 (1994).

[Minker 87] Minker, J.: *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann (1987).

[Moore 85] Moore, R. C.: Semantical Considerations on Non-Monotonic Logic, *Artif. Intell.*, Vol. 25, pp. 75-94 (1985).

[西尾 93] 西尾章治郎：大規模データベースにおける知識獲得，情報処理，Vol. 34, No. 3, pp. 343-350 (1993).

[Shapiro 81] Shapiro, E. Y.: Inductive Inference of Theories from Facts, Research Report 192, Dept. Computer Science, Yale Univ. (1981).


〔担当編集委員・査読者：大貝晴俊〕

## ——著者紹介——


**井戸 譲治**

1992年大阪大学工学部通信工学科卒業．1994年同大学大学院工学研究科前期課程修了．同年，三菱電機(株)入社．産業システム研究所に勤務．在学中，論理に基づく知識表現および推論機構の研究に従事．情報処理学会会員．

**馬場口 登**(正会員)

1979年大阪大学工学部通信工学科卒業．1981年同大学院前期課程修了．1982年愛媛大学工学部助手．大阪大学工学部助手，講師を経て，現在，大阪大学産業科学研究所助教授．工学博士．人工知能，パターン認識，画像処理の研究に従事．IEEE，電子情報通信学会，情報処理学会各会員．

**北橋 忠宏**(正会員)

1962年大阪大学工学部通信工学科卒業．1968年同大学院博士課程修了．同年，大阪大学基礎工学部助手．同助教授，豊橋技術科学大学助教授，教授を経て，1986年大阪大学産業科学研究所教授．工学博士．3次元物体認識のための視覚システム，自然言語処理，学習・推論機構に関する研究に従事．本学会元理事，IEEE，電子情報通信学会，情報処理学会，日本認知科学会，計量国語学会各会員．