人　工　知　能　74－2
（1991．1．16）

自己認識的データベースについて

森 有一　　馬場口 登　　手塚 慶一

大阪大学　工学部

本稿で提案する自己認識的データベースAEDBは，非単調論理の一つである自己認識論理の意味論に基づき，論理プログラムの証明手続きを利用したデータベースである．AEDBは自己認識推論を行うことにより，演繹データベースでは不可能であった不完全な知識に対する問い合わせを実現している．AEDBは，一階述語形式で書かれた知識を，対象とする世界を制限し，命題自己認識論理式の集合と捉えることにより，一階自己認識論理の非決定性を回避する．またAEDBは，形式的な証明法を用いることにより，効率の良い質問応答を実現し，問い合わせの事項を満足するオブジェクトを検索する機能も併せ持つ．

On Autoepistemic Databases

You-ichi Mori　Noboru Babaguchi　Yoshikazu Tezuka

Faculty of Engineering,Osaka University

2-1 Yamadaoka,Suita-shi,Osaka,565 Japan

In this paper, we propose an AutoEpistemic DataBase AEDB *which is based on autoepistemic logic that is one of non-monotonic logics.* AEDB *is capable of dealing with integrated knowledge base of complete and incomplete information. Some assumptions on* AEDB*'s domain avoids undecidability of first order autoepistemic logic by considering AEDB expressions as propositional formulas.* AEDB *can answer queries efficiently for its proof-theoritic procedure.*

1．まえがき

近年，論理型言語の発展と相まって，演繹データベース［Mink87］［勝野90］への関心が高まっている．演繹データベースは，事実を意味する個々のデータと，データの関係に関する知識から構成され，その関係を使ってデータベースには直接記述されていないデータを演繹推論により導き出すことができる．しかし演繹推論は，完全な知識のみを対象とし，例外を含む知識などの不完全な知識を扱うことができない．よって演繹推論を越える推論方式を導入することが，データベースを高度化する重要な鍵の一つとなろう．

不完全な知識をも扱いうる推論方式の一つに非単調推論がある．非単調推論は，公理の増加による定理集合の増加が非単調であるという，論理的な特徴を持つが，これは，人間が不完全な知識から推論を行うときの“信念の翻意（所有する信念を新しい知識の追加により覆す）”とよく似ている．よって非単調推論は，人間の行う推論を実現する有力な手法として注目を集めている．

非単調推論を定式化する非単調論理の一つである自己認識論理[Moore85]に基づき，不完全な知識をも扱いうる非単調推論システムの一つに，筆者らの提案したＢＭＳ［森馬91］がある．ＢＭＳは完全な知識だけではなく，“⇐”を用いて表現される不完全な知識を受け取り，それらをもとに信念に基づく推論を行い，妥当な信念集合を得るという推論システムである．

“⇐”は例外許容型オペレータであり，これを用いることにより，様相記号を含まない簡単かつ直感的な知識表現が可能となる．しかしＢＭＳの推論手続きは，可能世界意味論に基づくモデル論的なものであるため，知識表現に出現する命題定数の増加にともない，推論に必要な計算量が指数関数的に増加し，大規模なシステムへの応用には問題がある．

本稿では，ＢＭＳの命題形式の知識表現を一階述語形式に拡張した自己認識的データベースＡＥＤＢ（AutoEpistemic DataBase）を提案する．ＡＥＤＢは，対象とする世界を制限することにより，一階自己認識論理式の非決定性を回避し，推論を可能とする．またＡＥＤＢは，その知識表現をプログラムと見なし，証明論的な推論手続きを用いて，効率の良い，質問応答を実現している．また質問事項を満足するようなオブジェクトを検索する機能も併せ持つ．

本稿では，まずはじめにＡＥＤＢの論理的根拠となる自己認識論理について説明し，次いでＡＥＤＢの知識表現，対象とする世界，質問応答手続きについて述べ，最後にＡＥＤＢの実行例について説明を加える．

2．自己認識論理の概要

ＡＥＤＢは自己認識論理の表現形式・意味論を利用している．自己認識論理は古典的論理の体系に様相記号Ｌを加え拡張することによって，先に得られた結論が無効となりうる非単調推論を定式化した論理である．自己認識論理によって定式化される推論は，自己認識推論と呼ばれ，自分自身の信念の状態を把握しているエージェントの推論をモデル化したものである．

さて自己認識論理における重要な概念に安定拡張（stable extension）［Moore85］がある．与えられた自己認識論理式集合から，その安定拡張を求めることは，自己認識推論を行うことと同値であり，安定拡張は，自己認識推論による推論結果と考えることができる．

《定義２．１》（安定拡張）［Moore85］
前提と呼ぶ自己認識論理式集合をＡ，Ａの安定拡張をＴとする．Ｔは以下の条件を満足する自己認識論理式の集合である．

$$T = Th(A \cup \{L\tau \mid \tau \in T\} \cup \{\neg L\tau \mid \tau \notin T\})$$

Thは，標準論理における定理集合を求めるオペレータとする．■

以下，本文では，ギリシャ文字は論理式を表す．

Konoligeは，Mooreの提案した安定拡張が，前提とは弱い依存関係しか持たないことを指摘し，推論の結果として人間の直感にそぐわない場合があることを示した［Kono88］．また前提と強い依存関係を持つ，より合理的な概念である強依存拡張（strongly grounded extension）を提案した．［Kono88］では，Mooreの安定拡張は，弱依存拡張（weakly grounded extension）と呼ばれている．

《定義２．２》（強依存拡張）［Kono88］［松本89］
α，$\beta_1, \ldots, \beta_n$，γを，通常式と呼ばれるＬを含まない式とする．
$\gamma \leftarrow L\alpha \wedge \neg L\beta_1 \wedge \ldots \wedge \neg L\beta_n$のような形をした自己認識論理式の前提集合をＡ，Ａの強依存拡張をＴとする．Ｔは以下の条件を満足する自己認識論理式の集合である．

$$T = Th_{ss}(A' \cup \{L\tau \mid \tau \in A'\}$$

$$\cup \{\neg L\tau \mid \tau \notin T_0\})$$

ただし，T_0は，Tに含まれる通常式の集合，

$$A' = A - \{\gamma \leftarrow L\alpha \wedge \neg L\beta_1 \wedge \ldots \wedge \neg L\beta_n \mid \beta_i \in T(i=1,\cdots,n)\}.$$

安定集合（stable set）T_{stable}は，以下の(1)～(3)の条件を満足する自己認識論理式の集合である．

(1) $Th(T_{stable}) = T_{stable}$

(2) $\{L\alpha \mid \alpha \in T_{stable}\} \subseteq T_{stable}$

(3) $\{\neg L\alpha \mid \alpha \notin T_{stable}\} \subseteq T_{stable}$

尚，自己認識論理式の集合Tに対し，

$$Th_{ss}(T) \triangleq Th(T_{stable}) \quad \blacksquare$$

ＡＥＤＢは，「ｐである」のような事実（ファクト）や，「ｐならばｑである」のようなルールで表される完全な知識に加え，「ｐならば普通ｑである」のような不完全な知識であるデフォルトをも対象とする．またこれらの知識を述語形式で表現することにより，演繹データベースの特色であるデータベースに記述されていない新しいデータを導き出す推論能力を持つ．しかし，一階自己認識論理の弱依存拡張や強依存拡張を求めるとき，定義から分かるように，論理的帰結集合に含まれない論理式をすべて求める必要がある．これは一階述語論理の非決定性より，無限のプロセスを必要とすることが知られ，弱依存拡張や，強依存拡張を求める有限回の手続きが存在しない．すなわち，一階自己認識論理の非決定性である．

３．自己認識的データベースＡＥＤＢ

ＡＥＤＢでは，命題自己認識論理が決定的であることに着目し，演繹データベースで一般に仮定されている制限を用いることにより，一階自己認識論理式で記述された知識を等価な命題自己認識論理式に置き換え，非決定性問題を回避する．ここでは，まずＡＥＤＢの知識表現を定義し，次いでＡＥＤＢの対象とする世界を明らかにし，ＡＥＤＢの知識表現の意味を，命題自己認識論理式を用いて与える．

３．１　ＡＥＤＢの知識表現

ＡＥＤＢでは，ファクト（事実），ルール（規則）といった完全な知識に加え，デフォルトという不完全な知識をも扱う．ここでは，ＡＥＤＢの扱う知識表現について詳述する．

【ファクト】

「Spankyは鳥である」といった，事実（個体に関する明示的な知識）をbird(Spanky)といった一階述語のリテラルで表す．

《定義３．１》（ファクト式）

$$p(a_1, \ldots, a_n)$$
$$\neg p(a_1, \ldots, a_n)$$

のような，一階述語の正または負の基礎リテラル．ただし，$a_1, \ldots, a_n$は定数，¬は論理的否定を表す．また上式はファクト式と呼ばれる． ■

このようなファクト式の集合からなるデータベースをＥＤＢ（Extensional DataBase）と呼ぶ．

【ルール】

「Xがペンギンであるならば，Xは鳥である」といった，個体と個体の間に成立する完全な知識をルールと呼ぶ．

《定義３．２》（ルール式）

ルール式とは，以下のような式である．

$$q \leftarrow p_1, \ldots, p_n.$$

ただし，$p_1, \ldots, p_n$，ｑは，正または負のリテラルであり，述語の引数は，全称限定された変数または定数のみである． ■

この式の直感的な意味は，「Xがp_1～p_nであるならば，Xはｑである」である．またこのようなルール式の集合からなるデータベースをＩＣＤＢ（Intensional Complete DataBase）と呼ぶ．

【デフォルト】

「Xが鳥であるならば，通常Xは飛ぶ」といった，個体と個体の間に成立する，例外を含む不完全な知識をデフォルトと呼ぶ．

《定義３．３》（デフォルト式）

デフォルト式とは，以下のような式である．

$$q \Leftarrow p_1, \ldots, p_n.$$

ただし，$p_1, \ldots, p_n$，ｑは，正または負のリテラルであり，述語の引数は，全称限定された変数または定数のみである． ■

この式の直感的な意味は，「Xがp_1～p_nであるならば，通常Xはｑである」である．

このようなデフォルト式の集合からなるデータベースをＩＩＤＢ（Intensional Incomplete DataBase）と呼ぶ．

３．２．ＡＥＤＢの対象とする世界

ＡＥＤＢは以下の方法で対象とする世界を制限する［Mink87］．

1) 領域閉包公理ＤＣＡ（Domain Closure Axiom）
ＤＣＡは変数の領域を，出現する全ての定数の集合に制限する公理である．ＡＥＤＢに出現す

る全ての定数の集合を $\{a_1, \cdots, a_n\}$ とすると，

$$DCA: \forall X\ (X=a_1) \vee \cdots \vee (X=a_n)$$

である（＝は等号理論ＥＱを満たす述語記号）．

2) ファンクションフリーである．

1)，2)より全ての述語の全ての引数は，定数もしくは領域を有限個の定数に制限された変数となる．この制限により，ファクト式，ルール式，デフォルト式を，論理的に同値な有限個の基礎式（すべての述語のすべての引数が定数であるような式）の集合で表すことができる．このように全称限定された変数を持つような式を，変数の領域内の定数を代入することによって生成されるすべての式に変換する．例えば，変数の領域を $\{a, b, c\}$ とすると，$\forall X\ p(X) \leftarrow q(X)$ は，$\{p(a) \leftarrow q(a),\ p(b) \leftarrow q(b),\ p(c) \leftarrow q(c)\}$ と変換される．

3) 単一名仮説ＵＮＡ（Unique Names Assumption）

ＵＮＡはすべての定数は違うという仮説であり，

$$UNA: \{a \neq b, a \neq c, \cdots, c \neq d, \cdots\}$$

である．

ＵＮＡを仮定すると，全ての基礎リテラルは区別できるので，例えば $p(a) \neq p(b)$ のように，各々の基礎リテラルを相異なる命題定数と見なすことができる．

以上のような制限の下では，ファクト式は命題定数，ルール式は，命題論理式の集合，デフォルト式は，命題自己認識論理式の集合と見なすことができる．これらの対象とする世界の制限は，直感的には，データベースに出現しないオブジェクトを無視し，オブジェクトどうしは互いに異なると見なすことに相当する．

よって，ＡＥＤＢに出現する定数を基礎代入することによって得られる，ＡＥＤＢの各式に対応する命題自己認識論理式の集合の強依存拡張を，ＡＥＤＢの推論結果と見なすことができる．これをＡＥＤＢ拡張と呼ぶ．

さて，命題自己認識論理は決定的であり，強依存拡張を求める手続きが存在する．従って，ＡＥＤＢ拡張を求める手続きが存在する．次章では，その手続きについて述べる．

４．ＡＥＤＢの推論手続き

ＡＥＤＢは，質問応答を，ＡＥＤＢ拡張を参照することにより行う．３章での考察より，ＡＥＤＢ拡張を求めることは，命題自己認識論理式の強依存拡張を求めることに等しい．

命題自己認識論理に基づく非単調知識処理システムＢＭＳは，可能世界意味論に基づくモデル論的な手法を用いて弱依存拡張を求めることにより自己認識推論を行う．しかし，式集合に含まれる命題定数の増加にともない，必要な計算量が指数関数的に増加し，大規模なデータベースには不向きである．

演繹データベースで広く用いられている論理プログラムは，ＳＬＤ導出と呼ばれる効率の良い形式的な証明手続きを持つ．ＳＬＤ導出は，一階述語論理式を対象とした形式的な証明手続きであり，対象となる式集合の増加が，計算量の指数関数的増加の問題を引き起こさない．また$P(X)$のような変数を含む証明を行うことで，Ｐを満足するような定数ａを検索することができる．

本稿では，ＡＥＤＢをある種の論理プログラムとみなし，ＳＬＤ導出によって構成されるＳＬＤ導出木を拡張した推論木（reasoning tree）を構成し，その推論木を参照することにより，質問応答を行う．

４．１質問応答手続き

まずプログラムである，ＡＥＤＢプログラムを定義する．

《定義４．１》（ＡＥＤＢプログラム）

以下のように定義された(1)～(3)のプログラムを総称してＡＥＤＢプログラムと呼ぶ．

(1) ファクトプログラムＦＰ（Fact Program）は，ＥＤＢに含まれるファクト式の集合である．

(2) ルールプログラムＲＰ（Rule Program）は，ＩＣＤＢに含まれる，すべてのルール式

$$q \leftarrow p_1, \cdots, p_n.$$

について，そのすべての対偶を表すルール式

$$q \leftarrow p_1, \cdots, p_n.$$
$$\neg p_n \leftarrow \neg q, p_1, \cdots, p_{n-1}.$$
$$\neg p_{n-1} \leftarrow \neg q, p_1, \cdots, p_{n-2}, p_n.$$
$$\vdots$$
$$\neg p_1 \leftarrow \neg q, p_2, \cdots, p_n.$$

の集合である．

(3) デフォルトプログラムＤＰ（Default Program）は，ＩＩＤＢに含まれる，すべてのデフォルト式の集合である． ■

ＳＬＤ導出は，プログラムの中から証明に用いる式を選ぶとき，ホーン節のみを対象とした論理プログラムでは，プログラム節を節形式で表すと，ヘッド（←の左側のリテラル）のみが，正のリテラルとなるので，ヘッドのみを参照する．ところがＩＣＤ

Bに含まれるルール式は，非ホーン節をも含むので，ルール式を節形式で表すと，複数個の正のリテラルが出現する．よってヘッドのみを参照する手続きを採用するには，ルール式の対偶を考慮する必要がある．このためRPに，ルール式のすべての対偶を含める．

一方デフォルト式q⇐pは，自己認識論理式を用いてq←Lp∧¬L¬qと表され，その対偶は(¬Lp∨L¬q)←¬qであるが，¬qが強依存拡張に含まれるとき，定義2．2のA'の定義より，¬qが含まれる強依存拡張にはq⇐pは含まれない．このため対偶を考慮する必要がない．

このように定義されたAEDBプログラムから，推論を行うため，推論木を構成する．推論木は，ノード（node）と，ノードとノードを結ぶブランチ（branch）からなり，推論のための探索空間を表す．以下に推論木の定義を示す．

《定義4．2》（ノード）

ノードとは，以下のようなリテラルの集合の対である．

$$\langle \{p_1, \cdots, p_n\}, \{q_1, \cdots, q_m\} \rangle$$

ただし，$p_1, \cdots, p_n$は前提列（premise list），$q_1, \cdots, q_m$は正当化列（justification list）と呼ばれる．また正当化列が空集合であるようなノードをゴールと呼ぶ．■

ノードの前提列は，AEDB拡張に含まれることを示さなければならないリテラルの集合である．それが示されたリテラルは，前提列から削除されていく．ノードの正当化列は，AEDBの拡張に含まれないことを示さなければならないリテラルの集合である．それが示されたリテラルは，正当化列から削除されていく．ゴールは，論理プログラムのゴール節に相当するものである．AEDBに質問するとき，前提列が，質問するリテラルからなるゴールを生成する．

次に，推論木の定義を述べる．本定義中での，単一化，最汎単一化子は，論理プログラムにおけるSLD導出のそれと同じものである．詳しくは，[Lloyd84] を参照されたい．

《定義4．3》（推論木）

以下の条件を満足するグラフを推論木と呼ぶ．

(a) 木の根のノードは，ゴールである．

(b) $\langle \{p_1, \cdots, p_n\}, \{q_1, \cdots, q_m\} \rangle$ を木のあるノードとする．$p \in$ EDBなるpとp_1が単一化可能であるとき，このノードは子孫をもつ．子孫は，

$$\langle \{(p_2, \cdots, p_n)\theta\}, \{(q_1, \cdots, q_m)\theta\} \rangle$$

であり，θはp_1とpの最汎単一化子である．

(c) $\langle \{p_1, \cdots, p_n\}, \{q_1, \cdots, q_m\} \rangle$ を木のあるノードとする．$p \leftarrow r_1, \cdots, r_j \in$ ICDBなるpとp_1が単一化可能であるとき，このノードは子孫をもつ．子孫は，

$$\langle \{(r_1, \cdots, r_j, p_2, \cdots, p_n)\theta\}, \{(q_1, \cdots, q_m)\theta\} \rangle$$

であり，θはp_1とpの最汎単一化子である．

(d) $\langle \{p_1, \cdots, p_n\}, \{q_1, \cdots, q_m\} \rangle$ を木のあるノードとする．$p \Leftarrow r_1, \cdots, r_j \in$ IIDBなるpとp_1が単一化可能であるとき，このノードは子孫をもつ．子孫は，

$$\langle \{(r_1, \cdots, r_j, p_2, \cdots, p_n)\theta\}, \{(\neg p_1, q_1, \cdots, q_m)\theta\} \rangle$$

であり，θはp_1とpの最汎単一化子である．

(e) $\langle \phi, \{q_1, \cdots, q_m\} \rangle$ を木のあるノードとする．$\langle \{q_1\}, \phi \rangle$ のような質問ゴールを根にもつ推論木の子孫のすべてのノードが失敗するとき，このノードは子孫をもつ．子孫は $\langle \phi, \{q_2, \cdots, q_m\} \rangle$ である．

(f) プログラムの各式には番号が与えられ，ノードから新たに(c)，(d)を用いて子孫のノードを生成する際，それらのノードを結ぶブランチには，そのとき用いられた式の番号を記入する．

(g) 手続き(c)，(d)，(e)，で子孫のノードを生成する際，そのノードの前提列と，その祖先のノードの前提列が単一化可能ならば，その祖先とその子孫のノードとの間のブランチに記入された番号のプログラム節は，単一化には使用できない．

(h) ノードが $\langle \phi, \phi \rangle$ となったとき，そのノードを成功ノードという．成功ノードは子孫をもたない．■

本手続きは，質問として与えられるリテラルが，AEDBプログラムの強依存拡張に含まれるかどうかを調べることにより，推論を行う．以下に推論木の直感的な説明を加える．また以下の説明において，“強依存拡張に含まれる”ことと“証明される”ことは同値である．

まず，Lp_iが証明されることと，p_iが証明されることの同値性を示す．p_iが証明されたとき，強依存拡張は，安定集合であるので，Lp_iが証明される．またその逆も成り立つ．その理由を以下に示す．

p_iが証明されないとき，Lp_iが証明されたと仮

定する．するとLp_iは，強依存拡張が安定集合であるための条件から証明されるのではなく，ＡＥＤＢに含まれる式から導き出されなければならない．このためには，ＡＥＤＢの知識表現に用いられる式の制限より，$\neg p_i \Leftarrow q$のような，⇐の左側に$\neg p_i$が出現するような式が含まれなければならない．これを自己認識論理式の節形式に変換すると，

$$Lp_i \vee \neg Lq \vee \neg p_i$$

となる．ところが，p_iと$\neg Lp_i$は，安定集合の条件より，同時に存在しえないので，この式からLp_iは証明されない．よって上の仮定は矛盾し，p_iが証明されないとき，Lp_iも証明されない．よってLp_iを証明することは，p_iを証明することと同値である．

ノード〈$\{p_1, \cdots, p_n\}$，$\{q_1, \cdots, q_m\}$〉は，ゴール節

$$\leftarrow Lp_1 \wedge \cdots \wedge Lp_n \wedge \neg Lq_1 \wedge \cdots \wedge \neg Lq_m$$

を意味する．プログラムに$p_1 \leftarrow r_1 \wedge \cdots \wedge r_k$が含まれているならば，$Lp_1$と$p_1$の同値性より，上のゴール節と$p_1 \leftarrow r_1 \wedge \cdots \wedge r_k$の論理的帰結として，

$$\leftarrow r_1 \wedge \cdots \wedge r_k \wedge Lp_1 \wedge \cdots \wedge Lp_n \wedge \neg Lq_1 \wedge \cdots \wedge \neg Lq_m$$

が得られる．よってLr_iとr_iの同値性より，

$$\leftarrow Lr_1 \wedge \cdots \wedge Lr_k \wedge Lp_1 \wedge \cdots \wedge Lp_n \wedge \neg Lq_1 \wedge \cdots \wedge \neg Lq_m$$

が得られる．これは，定義４．３の(c)により，子孫のノードを生成することに相当する．

プログラムに$p_1 \Leftarrow r_1 \wedge \cdots \wedge r_k$が含まれているならば，プログラムに

$$p_1 \leftarrow Lr_1 \wedge \cdots \wedge Lr_k \wedge \neg L \neg p_1$$

が含まれていることと同値であり，同様に

$$\leftarrow Lr_1 \wedge \cdots \wedge Lr_k \wedge Lp_1 \wedge \cdots \wedge Lp_n \wedge \neg Lq_1 \wedge \cdots \wedge \neg Lq_m \wedge \neg L \neg p_1$$

が得られる．これは，定義４．３の(d)により，子孫のノードを生成することに相当する．

$\neg Lq_i$の証明は，q_iの証明が失敗することによって示される．これは論理プログラムのnegation as failure rule の概念を自己認識論理式に応用したものであり，定義４．３の(e)により，正当化列が変更されていくことを意味する．

定義４．３の(g)の直感的な意味は，同じリテラルを証明するために，同じ式が複数回使われるのを防ぐ制限である．

以上より成功ゴール〈ϕ，ϕ〉を得たとき，上のゴール節の論理的帰結が空集合であり，したがって推論木のゴールが成功したことを意味する．

次に，ＡＥＤＢの質問応答手続きについて述べる．質問応答は，質問を意味するゴールから生成される推論木を参照することにより行われる．

【質問応答手続き】

$p_1 \wedge \cdots \wedge p_n$が推論されるかどうかを調べるとする．

１．ゴール〈$\{p_1, \cdots, p_n\}$，ϕ〉を生成する．

２．ゴール〈$\{p_1, \cdots, p_n\}$，ϕ〉に対する推論木を構成する．

３．(1) ゴールに変数が含まれていないとき

(a) 成功ノードが存在しないとき

“NO”と答える．

(b) 成功ノードが存在するとき

ゴールから，成功ノードまでのブランチに，ＤＰに含まれるプログラム節の番号がつけられていない，成功ノードが存在するならば，“TRUE”と答える．そうでなければ，“MAYBE TRUE”と答える．

(2) ゴールに変数が含まれるとき

(a) 成功ノードが存在しないとき

“NO”と答える．

(b) 成功ノードが存在するとき

ゴールから，成功ノードまでのブランチに，ＤＰに含まれるプログラム節の番号がつけられていない成功ノードのすべてに関して，ゴールに含まれる変数へ行われた代入を示すとともに，“TRUE”と答える．そうでない成功ノードに関しては，ゴールに含まれる変数へ行われた代入を示すとともに，“MAYBE TRUE”と答える．

質問応答手続きにおいて，変数を含まないリテラルに関して質問することは，そのリテラルがＡＥＤＢプログラムの強依存拡張に含まれるかどうかを調べることに相当する．デフォルト式を用いず証明が成功したならば，完全な知識のみを用いて証明が成功したことになり，得られた推論結果は完全な知識と考えられ，“TRUE”（完全に正しい）と答える．デフォルト式を用いた場合は，不完全な知識をも用いて証明されているので，得られた推論結果は，新しいデータの追加などにより，覆される可能性がある．よって“MAYBE TRUE”（たぶん正しい）と答える．

変数を含むリテラルに関して質問することは，質

問のリテラルと単一化可能かつ，証明が成功するような基礎リテラルを検索することである．本手続きでは，完全な知識のみを用いて証明がなされた基礎リテラルと，不完全な知識をも用いて証明された基礎リテラルとを，“TRUE”と“MAYBE TRUE”に区別して答える．

5．実行例

ここでは，実例を挙げ，ＡＥＤＢの質問応答について説明を加える．

ＡＥＤＢに以下の知識を与える．

ＥＤＢ＝ {bird(Ted), bird(May), penguin(Tom), move(May), move(Tom)}

ＩＣＤＢ＝ {bird(X)←penguin(X), ¬fly(X)←penguin(X)}

ＩＩＤＢ＝ {fly(X)⇐bird(X), alive(X), alive(X)⇐move(X)}

上のような知識を，ＡＥＤＢプログラムに変換する．番号付けがなされたＦＰ，ＲＰ，ＤＰは以下のようになる．

ＦＰ＝ {bird(Ted), …①
bird(May), …②
penguin(Tom), …③
move(May), …④
move(Tom)} …⑤

ＲＰ＝ {bird(X)←penguin(X), …⑥
¬penguin(X)←¬bird(X), …⑦
¬fly(X)←penguin(X), …⑧
¬penguin(X)←fly(X)} …⑨

ＤＰ＝ {fly(X)⇐bird(X), alive(X), …⑩
alive(X)⇐move(X)} …⑪

ＡＥＤＢに対し，fly(X)？という質問を与え，検索を行う．図1は，fly(X)という質問から生成される推論木を示したものである．ここでは，X=Mayが導かれる成功ブランチについてのみ説明する．

1．〈 {fly(X)} , φ〉というゴールが，木の根のノードとなる．

2．fly(X)は，⑩のfly(X)⇐bird(X), alive(X)のfly(X)と単一化可能であるため，定義４．３の(d)より，
〈 {bird(X), alive(X)} , {¬fly(X)} 〉が生成される．

3．bird(X)は，②のbird(May)と最汎単一化子θ＝{X/May} により単一化可能であるため，定義４．３の(b)より，
〈 {alive(May)} , {¬fly(May)} 〉が生成される．

4．alive(May)は，⑪のalive(X)⇐move(X)のalive(X)と最汎単一化子θ＝{X/May} により単一化可能であるため，定義４．３の(d)より，
〈φ, {¬fly(May), ¬alive(May)} 〉が生成される．

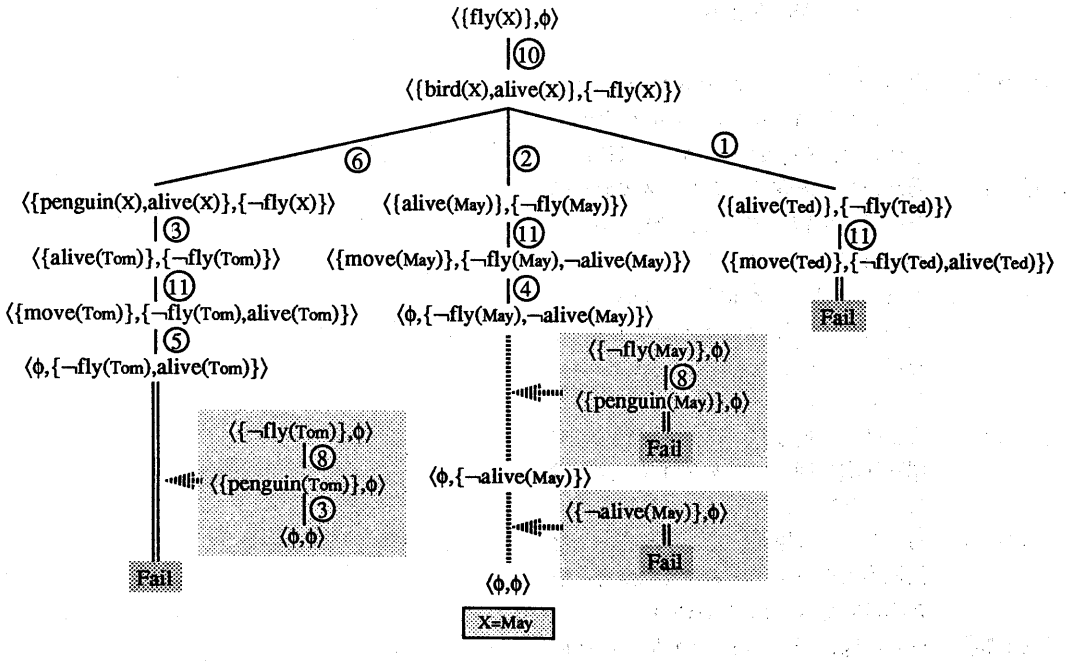

図1. 推論木

5．ゴール 〈 {¬fly(May)} , ϕ〉 により構成される推論木は，成功ブランチを持たないので，定義４．３の(e)を用いて，ノードの正当化列から¬fly(May)を削除する．

6．ゴール 〈 {¬alive(May)} , ϕ〉 により構成される推論木は，成功ブランチを持たないので，定義４．３の(e)を用いて，ノードの正当化列から¬alive(May)を削除する．

7．定義４．３の(h)より，成功ノード〈ϕ, ϕ〉が得られ，このノードは成功し，同時にX=Mayを得る．

６．むすび

本稿では，自己認識論理に基づく自己認識的データベースＡＥＤＢを提案し，不完全な知識に対する質問応答を実現した．ＡＥＤＢは，対象とする世界を制限することにより，一階自己認識論理式の非決定性を回避し，推論を可能とした．またその知識表現をプログラムと見なし，論理プログラムのＳＬＤ導出を拡張した証明論的な推論手続きを与え，効率の良い質問応答を実現している．

今後は，論理プログラムの意味と自己認識論理との関係［Przy88］を考慮し，ＡＥＤＢの宣言的・手続き的意味論について考察を進め，その論理的性質を明らかにしていく予定である．

参考文献

［Mink87］ Minker, J. : Foundations of Deductive Databases and Logic Programming, Morgan Kaufmann (1987).

［勝野90］ 勝野裕文：演繹データベースの形式的意味論，情報処理，Vol. 31, No. 2, pp. 198-205 (1990).

［Moore85］ Moore, R. C. : Semantical Considerations on Nonmonotonic Logic, Artif. Intell., Vol 25, pp. 75-94 (1985).

［森馬91］ 森馬純一，馬場口 登，手塚慶一：非単調知識処理システムＢＭＳ，情処論文誌，Vol. 32, No. 1 (1991).

［森有90］ 森 有一，馬場口 登，手塚慶一：自己認識的データベースの基礎検討，情処第41回全大，5L-9 (1990).

［Kono88］ Konolige, K. : On the Relation between Default and Autoepistemic Logic, Artif. Intell., Vol. 35, pp. 343-382 (1988).

［松本89］ 松本裕治，佐藤 健：非単調論理と常識推論，情報処理，Vol. 30, No. 6 pp. 674-683 (1989).

［Przy88］ Przymusinski, T. C. : On the Relationship Between Logic Programming and Non-monotonic Reasoning, Proc. AAAI-88, pp. 444-448 (1988).

［Lloyd84］ Lloyd, J. W. : Foundations of Logic Programming, Springer-Verlag (1984).