解　説

人工知能とデータ・ベース*

古　川　康　一**

1. はじめに

近年の人工知能の研究は，知識の表現，論理的な演繹，あるいは人間が日常行っている常識的な推論などに焦点があてられ，限られた問題領域で知的なふるまいをするシステムが開発されてきた．そこでは，知識の集積としてのデータ・ベースは高度の機能を要求され，それがデータ・ベース処理技術の発展を促す原動力となった．本稿では，まず人工知能システムにおけるデータ・ベースの役割りから，そこでの重要な機能を明らかにし，それらの機能がどのように実現されているかを見ることにしよう．

ここでは，人工知能システムとして，高度な質問応答システムを考える．これは，この話題がデータ・ベースに関心のある読者の興味からそれほど外れないであろうと思われることと，知識データ・ベースの概略を論ずるには，それが最も適していると思われるからである．このシステムでは，ユーザの質問に対して，データ・ベースから答を引き出す操作は，単純なデータの検索のみに留まらず，人間の思考過程に対応する処理も含まれる．それは推論と呼ばれる過程で，それによってデータ・ベース中に個別的事実として含まれていないような事実をも引き出すことができるわけである．

人間の行う推論には，演繹，帰納，類推などがあるが，本稿では，これらの各機構の詳細について述べるのが目的ではなく，知識としてのデータの表現，その取り扱いなどを考える上では，その内で，演繹のみを考察すれば十分であろう．特に演繹のみを取り上げるのは，問題解決の場合で考えると，帰納，類推などは発見的な戦略としては非常に重要であるが，論理的に正しい解を得るには，演繹によらなければならないからである．

つぎに，思考あるいは演繹の過程をより詳細に調べてみると，論理が複雑になると一直線に解に到達できない．ある場合には，誤った思考に陥り，容易に解に到達し得ないこともある（筆者の経験によれば，ハノイの塔の問題で，塔が7段ないし8段になると，途中で道草せずに解に到達するのは非常に困難である）．このような，途中に試行錯誤を含む計算過程は，非決定性アルゴリズムとして記述される．

この処理を直線的な決定性アルゴリズムの形にはめ込むために考えられたのが後戻り制御（backtracking control）で，ある可能性が失敗したときに，他の可能性を調べる場合，あたかも失敗した部分を実行しなかったように見せかけるために，分岐点から失敗した時点までの間に行ったデータの変更をすっかり元の状態に戻す．解に到達した時点で，後戻り制御によって無効になった部分をのぞく計算過程をながめてみると，各分岐で正しい選択を与えて得られる決定性アルゴリズムを走らせた場合と全く同じ過程をたどっていることがわかる．

非決定アルゴリズムのもう1つの処理方法は，複数の可能性をコルティンによって同時に，なかば別々に調べていくやり方である．この方法は，データ・ベースに，同時に複数の状況を表現する機能を要求する．この擬似並列処理は，各々の処理が互いに連絡を取り合ったり，より高度の戦略によって実行の各ブランチの重要度を評価して実行順序を決定する機構を採り入れることが可能である．また将来，実際に並列プロセッサによる並列処理を導入することも可能であろう．

以上の考察から明らかなように，人工知能でのデータ・ベースに関連する処理技術の内，注目に値するのは，演繹に適した知識の表現，および同時に複数の状況を表現する機構の2つである．次章では，2つの全く異なる知識の表現形態，およびそれらの上で働く演繹システムの概略を紹介する．3.では，同時に複数の

* On Database in Artificial Intelligence by Koichi FURUKAWA (Computer Science Division, Electrotechnical Laboratory)
** 電子技術総合研究所ソフトウェア部情報システム研究室

状況を表現する機構について述べる．最後の章では，ここで述べた技術の実用化への考察，今後のこの分野での研究課題などについて述べる．

2. 知識の表現および演繹の機構

多くの知識から，論理操作によって新たな事実を引き出すのが演繹であるが，そこでの知識および論理操作のイメージをよりはっきりさせるために，簡単な例を示そう．

いま，「リンゴは赤い」という事実を考えてみよう（ここで，世の中には青いリンゴや黄色いリンゴもあることは考えない）．これは，つぎの意味で，抽象化された事実（以下，抽象的事実という）である．すなわち，いまリンゴ $a, b, c, \cdots$ があったとすると，それは「a はリンゴである」，「b はリンゴである」，「c はリンゴである」，…という個別的事実として，データ・ベースにおかれる．一方，「a の色は赤」，「b の色は赤」，「c の色は赤」，… も個別的事実であるが，これらは，このデータ・ベースと「リンゴは赤い」という抽象的事実から，論理的に導き出すことができる．すなわち，「リンゴは赤い」という命題は，「x がリンゴであれば，x の色は赤」といいなおすことができ，x が $a, b, c, \cdots$ の場合，この命題の仮定の部分が成り立つので，結論部分である「a の色は赤」，「b の色は赤」等が導き出せるわけである．

この例から容易にわかるように，データ・ベースを構成する知識には，個別的事実，および抽象的事実がある．また，抽象的事実は，含意命題「A ならば B」の形をとる．

ここでの論理操作は，三段論法 "A が真で，「A ならば B」であれば，B も真" によっているが，この操作を実行する計算手続きを考えると，「a はリンゴである」と「x がリンゴであれば，x の色は赤」の仮定部分の同一化がその中での基本演算である．

このような演繹操作を，広い範囲の問題に対して行うことのできるシステムを演繹システムと呼ぶことにする．演繹システムは，問題解決能力，すなわち，そのシステムが解き得る問題のクラスの大きさ，および各問題を解くのに要する時間（システムの効率）によって評価することができる．

演繹システムでの抽象的事実の表現方法には2通りある．その1つは，それを演繹プログラムのデータとして表わす方法で，他の1つは，それを演繹プログラムを構成する手続きとして表わす方法である．前者の表現を宣言的知識 (declarative knowledge) と呼び，後者の表現を手続き的知識 (procedural knowledge) と呼ぶ[1]．宣言的知識をとる演繹システムには，一階述語論理の演繹体系を基にしたシステムがあり，手続き的知識をとるシステムの例には，μ-PLANNER によって記述されているロボットの世界での質問応答システムがある[2]．

以下では，この2つの知識の表現方法およびそれに伴う演繹の機構の概略を説明しよう．

2.1 宣言的知識とその演繹機構

一階述語論理では，事実を表わすのに述語 (Predicate) と呼ばれる対象物の性質を表わす概念を用いる．たとえば，'a はリンゴである' は，リンゴであることを表わす述語 APPLE を用いて APPLE (a) と表わされる（ここで，述語は英字の大文字で表わし，対象物は小文字で表わすことにする）．また，「x がリンゴならば，x の色は赤」は，上の APPLE と，「赤い」を表わす述語 RED によって

$$\forall x\,(\mathrm{APPLE}(x) \Rightarrow \mathrm{RED}(x)) \qquad (1)$$

と表わされる．ここで，x は変数であり，$\forall x \cdots$ は「すべての x について…」を表わす．ここに現われた $\forall$ は限定子と呼ばれ，もう1つの限定子 $\exists$（$\exists x \cdots$ は，「…なる x が存在する」を意味する）と共に，この論理の表現能力を非常に豊かにしている（$\forall$ および $\exists$ を含まない論理として命題論理がよく知られているが，その論理では，「リンゴは赤い」などの抽象概念は表現できない）．

いま c がリンゴであるとき，「c は赤いか」という質問を考えてみる．データ・ベース中には，APPLE (a), APPLE (b), APPLE (c), … なる命題と，命題 (1) が存在する．質問は，論理式 RED (c) で表わされ，演繹システムは，この式を証明することを試みる．この式は，(1) を用いて，APPLE (c) が真ならば証明されることがわかるが，それはデータ・ベース中に個別的事実として存在するので，成り立つ．

ここで，このシステムの能力および問題点を明らかにするために，問題を少し複雑にしてみよう．いま，ある会社の従業員と組織の関係について考える．従業員は，計算機設計課 (Computer design section, cd-sect) に属し，その肩書きがシステムズ・エンジニア (se) であるとする．また，計算機設計課は計算機事業部 (Computer business department, cb-dept) の一下部組織とする．これらの事実は，述語 BELONG, TITLE, CONSTITUTE を用いて，それぞれ

BELONG (*a*, cd-sect)　(2)

TITLE (*a*, se)　(3)

CONSTITUTE (cd-sect, cb-dept)　(4)

と表わされる．つぎに，2つの抽象的事実を定義しよう．その1つは，所属関係 BELONG の組織の構造に対する推移律で，

$$\forall x \forall y \forall z\ (\text{BELONG}\ (x, y) \wedge \text{CONSTITUTE}\ (y, z) \Rightarrow \text{BELONG}\ (x, z)) \quad (5)$$

である．これは，x が y に属していて，y が z の一下部組織であれば，x は z に属していることを意味する．他の1つは，肩書きと仕事の内容を関連づけるもので，計算機事業部に属するすべてのシステムズ・エンジニアは計算機システムの設計 (computer systems design, cs-dsgn) を行っていることを表わすつぎのような論理式である．

$$\forall x\ (\text{BELONG}\ (x, \text{cb-dept}) \wedge \text{TITLE}\ (x, \text{se}) \Rightarrow \text{JOB}\ (x, \text{cs-dsgn})) \quad (6)$$

ここで，a の仕事が計算機システムの設計であること，すなわち JOB $(a,$ cs-dsgn$)$ を証明することはもちろん可能であるが，一歩進んで，a の仕事を求めることも可能である．その場合，質問は

$$\exists u\ \text{JOB}\ (a, u) \quad (7)$$

で表わされる．この論理式は真であることが示され，その証明の過程で u に cs-dsgn が代入されていることを知るのは容易である．この証明はつぎのようになされる．$\exists u$ JOB (a, u) を示すためには，(6)より，u を cs-dsgn として，BELONG $(a,$ cb-dept$) \wedge$ TITLE $(a,$ se$)$ を示せばよい．BELONG $(a,$ cb-dept$)$ は，(5)より BELONG $(a, y) \wedge$ CONSTITUTE $(y,$ cb-dept$)$ を満たす y が存在すればよいが，y=cd-sect とすれば，(2)および(4)によりこれは満たされる．ゆえに BELONG $(a,$ cb-dept$)$ は満たされる．一方，TITLE $(a,$ se$)$ は(3)で保証されているので，結局 $\exists u$ JOB (a, u) は証明され，u には cs-dsgn が代入されている．

一階述語論理は，妥当 (valid) なすべての命題を証明する演繹アルゴリズムを持っている．そのアルゴリズムとしては，自然演繹法と分解証明法とが知られているが[3]，分解証明法の方が，データ検索との対応がより自然である．そこでは，すべての個別的事実は変数を含まない公理として表わされ，すべての抽象的事実は，限定子を伴った含意公理として表わされる．

この演繹システムの問題点は，そのアルゴリズムが余りにも一般的なために，個々の問題に個有な戦略をとり込んだシステムに較べると効率が著しく悪いことである．このことを示すために，上の従業員の例に，つぎの2つの公理を追加してみよう．

$$\forall x\ (\text{BELONG}\ (x, \text{pb-dept}) \wedge \text{TITLE}\ (x, \text{se}) \Rightarrow \text{JOB}\ (x, \text{ps-dsgn})) \quad (8)$$

$$\forall x\ (\text{BELONG}\ (x, \text{cb-dept}) \wedge \text{TITLE}\ (x, \text{rs}) \Rightarrow \text{JOB}\ (x, \text{cs-rsch})) \quad (9)$$

ここで，pb-dept はプラント事業部，ps-dsgn はプラント・システム設計，rs は研究者 (research scientist)，cs-rsch は計算機システム研究を表わすものとする．そして，再び a の仕事を求めてみる．すると，$\exists u$ JOB (a, u) を示すためには，(6)，(8)，(9)のいずれかを使えばよいことはわかるが，それ以上は証明の過程が進んでからでないと分からない．ところがこの場合には，もし肩書きを先に調べられれば，その情報によって，用いるべき公理を選択できる．すなわち，もしそれが se であれば，(6)と(8)のみを考えればよく，もしそれが rs ならば(9)のみを考えればよい．このような個々の細かい検査を演繹システムに組み込むことによって，システムの効率は著しく向上することが予想される．このような反省から生れたのが，以下に述べる手続きによる知識の表現に基づいた演繹システムである．

2.2 手続き的知識とその演繹機構

手続き的知識を基にした演繹システムを実現するためには，その手続きを記述するためのプログラミング言語が必要となる．そのようなプログラミング言語には，近年の人工知能の研究に伴って開発された PLANNER[4]，μ-PLANNER[5]，CONNIVER[6]，QA4[7]，QLISP[8] などがあるが，ここでは QLISP によって説明を行う．

まず，その基本的な仕組みを明らかにするために，リンゴの例について考えよう．$A, B, C, \cdots$ がリンゴであることは，つぎの ASSERT 文により，データ・ベースに登録される．

(ASSERT (APPLE *A*))
(ASSERT (APPLE *B*))
(ASSERT (APPLE *C*))

ここで，括弧のつけ方，大文字の使用などは QLISP のシンタックスに従った．

つぎに(1)に対応する手続きを考える．それは，(RED X) という形の質問に対して (APPLE X) を探し出す手続きである．ここで大切な点は，質問 (RED *C*) が与えられたときに，この手続きが自動的に呼び出されることである．この手続き呼び出しのトリック

は，質問 (RED C) によるこの手続きの連想探索である．そのために，この手続きは見出しパターン (RED X) を持っている．この探索の際に質問と見出しパターンの照らし合わせが行われ，質問あるいは見出しパターン中に含まれる変数の値が定められる．この操作はこの節の初めに述べた同一化に相当するもので，ここではパターン整合と呼ばれている．

この手続きの本体は，GOAL 文

(GOAL (APPLE X))

である．この GOAL 文は，その引数と整合するデータをデータ・ベース中から探し出す．ここで，Xにはすでに値が与えられていることに注意を要する．この扱いは，(RED X) における X の扱いと異なっている．この相違を明確にするために，2種類の変数←X および \$$X$ を用いる．パターン整合に際して，←Xはすべての定数あるいは他の←変数と整合してその値を取り，\$$X$ はその時点ですでに与えられている値と同じ定数か←変数とのみ整合がとられる．すなわち，この手続きの見出しパターンは実際には (RED ←X) であり，本体は (GOAL (APPLE \$$X$)) である．この手続きの QLISP による述語は以下の通りである．

```
(R1 (QLAMBDA (RED ←X)
      (GOAL (APPLE $X))))                    (10)
```

ここで，R1 は関数名である．(10)の表現をよく見ればわかるように，この関数は LISP の LAMBDA 関数と同じ形をしており，QLAMBDA 関数と呼ばれる．

つぎに，従業員の問題を QLISP で表わしてみよう．(2)，(3)，(4)の事実は，**図-1**のようにASSERT文を実行することによって，データ・ベース中に置かれる．

所属関係の組織に対する推移律(5)は，つぎの QLAMBDA 関数 T1 で与えられる．

```
(T1
  [QLAMBDA
    (BELONG (←X←Z))
    (GOAL (BELONG ($X ←Y))
          APPLY $T1
    THEN (GOAL (CONSTITUTE ($Y $Z])            (11)
```

ここでGOAL文中の APPLY につづく \$T1 はデータ・ベース中に整合のとれるデータがない場合に呼び出される関数のクラスを示す．すなわち \$T1 の値は (T1)

```
6_(ASSERT (BELONG (A CD-SECT)
(BELONG (A CD-SECT)))
7_(ASSERT (TITLE (A SE)
(TITLE (A SE)))
8_(ASSERT (CONSTITUTE (CD-SECT CB-DEPT)
(CONSTITUTE (CD-SECT CB-DEPT))
```

図-1 ASSERT 文とその実行

である．また，THEN 以下は，(GOAL (BELONG (\$$X$ ←Y)) APPLY \$T1) の結果に対する検査を示し，もしこの検査に通らなければ，制御を再び前の GOAL に戻して，新たなデータを選び出させる．この制御が，前節で述べた後戻り制御である．

さて，つぎに(6)，(8)，(9)に相当する手続き群を考えよう．ここで，我々は，**2.1** の最後の部分の議論を思い起こす必要がある．すなわち，ここで与える手続き群は，そこに述べられている細かい検査が組み込まれている．

```
(J
  [QLAMBDA (JOB (←X ←Y))
  (ATTEMPT (IS (TITLE ($X SE)))
    THEN (CASES (JOB ($X $Y)) APPLY $JS1)
    ELSE (CASES (JOB ($X $Y)) APPLY $JS2])
                                               (12)
```

ここで，\$JS1 は (J1 J2) で，\$JS2 は (J3) である．すなわち，ATTEMPT 文によって，はじめに \$$X$ の TITLE が SE であるかどうか (IS) が調べられ，もしそうならば THEN 以下が実行され，そうでなければ ELSE 以下が実行される．ここで THEN, ELSE 中の CASES は，データ・ベースの探索を行わない点を除けば GOAL と同じである（ここで GOAL の代りに CASES を用いるのも，一種の発見的戦略と考えられる）．

```
(J1
  [QLAMBDA
    (JOB (←X CS-DSGN))
    (QPROG NIL
      (CASES (BELONG ($X CB-DEPT))
             APPLY $T1)
      (QRETURN (JOB ($X CS-DSGN])              (13)
(J2
  [QLAMBDA
    (JOB (←X PS-DSGN))
    (QPROG NIL
      (CASES (BBLONG ($X PB-DEPT))
             APPLY $T1)
      (QRETURN (JOB ($X PS-DSGN])              (14)
(J3
  [QLAMBDA
    (JOB (←X CS-RSCH))
    (QPROG NIL
      (CASES (BELONG ($X CB-DEPT))
             APPLY $T1)
      (QRETURN (JOB ($X CS-RSCH])              (15)
```

```
$1_(GOAL (JOB (A _U)) APPLY $J)
J:
#0= (JOB (A _U))
  J2:
  #0= (JOB (A _U))
    T1:
    #0= (BELONG (A CB-DEPT))
  J1:
  #0= (JOB (A _U))
    T1:
    #0= (BELONG (A CB-DEPT))
    (T1) = (CONSTITUTE (CD-SECT CB-DEPT))
  (J1) = (JOB (A CS-DSGN))
(J) = (JOB (A CS-DSGN))
(JOB (A CS-DSGN))
```

図-2 仕事を求める GOAL 文(16)の実行過程

質問文(7)は，つぎの GOAL 文で与えられる．

(GOAL (JOB (A ←U)) APPLY $J) (16)

ここで，$J の値はリスト (J) である．この GOAL 文によって関数 J が呼び出され，図-1 の 7 によって (TITLE (*A* SE)) がデータ・ベースに置かれているので，ATTEMPT の THEN 部分が実行され，その結果 J2 が先ず呼び出されるが，その実行は失敗し，つぎに J1 が呼び出される．J1～J3 の QRETURN の引数は，これらの関数の値を示し，ここでは答を返す仕掛けとして使っている．GOAL 文 (16) の実行過程を**図-2** に示す．

3. 複数の状況の同時表現

従来，データ・ベースはその時点で真であるデータの集積と考えられてきたが，ゲームで先の手を読んだり，経済予測をする場合などを考えると，いろいろな仮定に応じて異なったデータ・ベースを用意して，その上でデータを処理することが必要となる．その場合，仮定の相違によって生ずるデータの差異は，データ・ベース全体に及ぶとは考えにくい．多くの場合，ほんの一部分が変わるにすぎないであろう．このような状況を扱うために，データ・ベースの一部分を一時的に変更して，見方によって，変更前のデータ・ベースを見たり，変更されたデータ・ベースを見たりすることのできる方法が考え出された．それをここでは視点機構と呼ぼう（英語では context mechanism[6),8)] である)．

変更されたデータは，視点枠 (context frame) と呼ばれるもとのデータ・ベースと異なる場所に置かれる．1つのデータ・ベースには複数個の視点枠を置くことができ，それは**図-3** のように木構造によって互いに連結されている．視点は，任意の視点枠あるいはもともとのデータ・ベース（これも1つの視点枠と考えられ，global という名前で参照される）に置くことができる．いま，この図で，視点を cf_6 に置くと，**図-4** のように，cf_6-cf_4-cf_2-global と見渡せるわけである．

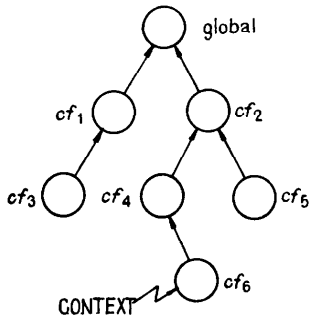

図-3 視点枠の木

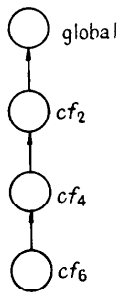

図-4 参照可能な視点枠とその順序

ここで，従業員の問題で *A* を仮に他の課に配置換えすることを考える．すると，global 中にある (BELONG (*A* CD-SECT)) を，たとえば cf_6 で否定しなければならない．ここで，global 中のデータ (BELONG (*A* CD-SECT)) は影響を受けてはならない．すなわち，視点を global に戻せば，このデータは真でなければならない．このためのトリックは，データの状態を表わす属性 MODELVALUE を，各視点枠に置くことができるようにすることで，いまの例でいえば，cf_6 での (BELONG (*A* CD-SECT)) の MODELVALUE を NIL（偽あるいは空を表わす LISP の定数）とすればよい．

この視点機構と並列プログラミング技術を組み合わせることによって，非決定性アルゴリズムを記述できることはすでに述べた．

この視点機構での視点枠の構造は，木構造であるが，これをルート・グラフ構造に拡張できる．すなわち，そこでは2つ以上の視点枠が，1つの視点枠を共通の子として持つことができる．この拡張された視点構造を取り入れたセマンティック・ネットワーク[9)]は，その中で一階述語論理の論理式を表わすことができ

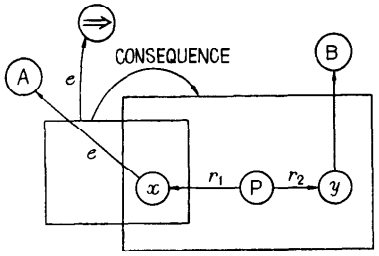

図-5 論理式 $\forall x\{[x\in A]\Rightarrow\exists y\{[y\in B]\wedge P(x,y)\}$ のセマンティック・ネットワークによる表現

る．そこでは，含意規則での仮定および結論を表わすのに別々の視点枠を用いている．また，視点枠自身をセマンティック・ネットワークでの節と考え，その2つの視点枠を弧 CONSEQENCE で結合している．**図-5** に，その表現例を示す．これは，知識の第3の表現形態と考えられ，この表現に基づいた演繹システムが開発されている．また，このセマンティック・ネットワークは，自然言語の内部表現としても使われており，それを用いた自然言語によるデータ・ベースへの質問応答システムの開発が計画されている[10]．

4. おわりに

これまで述べてきたように，演繹機構および視点機構のデータ・ベースへの導入は，人工知能研究によってもたらされた大きな成果であるといえよう．しかしながら，それらの機構は，主記憶に十分納まるぐらいの小規模なデータ・ベース上に実現されているに過ぎない．また，それらのデータ・ベースのほとんどは，複雑な処理に耐えるように LISP のリスト構造のような柔軟なデータ構造上に作られている．そこでは，大規模データ・ベースに対する考慮は一切なされていない．このことは，大量の知識を持つ人工知能システムの開発が困難なことと，この技術が商用のデータ・ベースには直接応用できないことを意味する．この大規模化を達成する方向には2通り考えられる．その1つは，現在のこれらのデータ・ベースの土台となっているリスト構造のレベルでの大規模化である[11),12)]．これは，リスト構造をファイル化し，2次記憶へ格納することを意味する．ここでの問題は2次記憶と主記憶間の I/O による効率の低下および文字情報 (LISP でのアトム名) の管理である．LISP では，主記憶内にアトム名の辞書を持ち，リスト中の要素として現われるアトムは，計算機の中ではこの辞書によって与えられる内部識別子によって表わされる．しかし大規模データ・ベースを実現するためには，このアトムの辞書自身を2次記憶上で管理しなければならない．

他の1つは，既存のデータ・ベースとの結合である．そのためには，外部のデータ・ベースの構造，意味などを，知識の一部分として知識データ・ベースに取り込み，その情報を利用してユーザの質問から外部データ・ベースへの検索要求を作り出す機構を演繹機構の一部分として持たせることが必要となる[10),13)]．この検索要求の作成時には，主鍵検索，逆引きファイルなどの索引の利用などを考慮した最適化処理がなされなければならない[14)]．

本稿では触れなかったが，PLANNER の demon の考えも興味のあるもので，データの追加，削除の際に無矛盾性を維持するための機構として使うことができる．それは，連想探索による関数呼び出しと，コルティン制御を組み合わせた方法で，特定のパターンを持ったデータの追加，削除が，データ・ベースの無矛盾性を維持するために一連の処理を行う，そのパターンに対応したコルティンを駆動する．

無矛盾性，あるいはより一般にデータ・ベースの統合性 (Integrity) は，それ自身非常に厄介な問題である．特に，意味内容に立ち入った統合性を達成するためには，データ・ベースの意味を取り扱わなければならない．この表現にセマンティック・ネットワークを用いた興味深い研究が報告されている[15)]．この取り組みは，先に述べた既存のデータ・ベースと知識データ・ベースとの結合と共に，人工知能とデータ・ベースの，本稿では取り上げなかった新たな接点，すなわちデータ・ベース自身の意味の取り扱いに根ざしており，今後，この分野での研究の急速な発展が期待される．

参考文献

1) Winograd, T.: Some thought on the declarative/Procedural Controversy. Selected Papers by Prof. Winograd, Project PIPS, ETL (March 1974).
2) Winograd, T.: Understanding Natural Language, Academic Press, New York (1972).
3) Manna, Z.: プログラムの理論，五十嵐滋訳．日本コンピュータ協会 (1974).
4) Hewitt, C.: Description and theoretical analysis (using schemata) of PLANNER: A language for proving theorems and manipulating models in a robot. AI Memo No. 251, MIT Project MAC (April 1972).
5) Sussman, G. J., Winograd, T. and Charniak, E.: MICRO-PLANNER Reference Manual,

MIT. AI Memo No. 203 A (Dec. 1971).
6) McDerwott, D. V. and Sussman G. J.: The CONNIVER reference manual. AI Memo No. 259, MIT Project MAC (May 1972).
7) Rulifson, J. F., Derksen, J. A., and Waldinger, R. W.,: QA4: A procedual calulus for intuitive reasoning. Artificial Intelligence Center. Technical Note 73, SRI (1972).
8) Reboh, R. and Sacerdoti, E.: A PRERIMINARY QLISP MANUAL. SRI AI Center Technical Note 81 (August 1973).
9) Hendrix, Gary G.: Expanding the Utility of Semantic Networks Through Partitioning. Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, pp. 115～121. Tbilisi, Georgia, USSR (Sept. 1975).
10) Sacerdoti, E.: SRI Internal Report on decision aids for command and control (1976).
11) 古川康一：人工知能応用のためのデータ・ベース．京都大学数理解析研予稿集 (1975年5月).
12) Clark, D. and Green, C. C.: An Empirical Study of List Structure in LISP. Selected Papers by Prof. Green, Project PIPS, ETL (August 1975).
13) 古川康一：New Formulation of Data Base Semantics. Hand written Memo (1976).
14) Raphael, B.: The Thinking Computer, pp. 173～174. W. H. Freeman and Company (1976).
15) Roussopoulos, N., Mylopoulos, J.: Using Semantic Networks for Data Base Managements. Proc. First International Conference on Very Large Data Bases, Boston (Sept. 1975).

(昭和51年6月7日受付)