ICOT Technical Memorandum: TM-0163

TM-0163

# メタプログラミングによる知識獲得支援のための基礎技術確立をめざして

國藤　進

April, 1986



ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

# メタプログラミングによる知識獲得支援のための基礎技術確立をめざして

國藤　進（ＩＣＯＴ）

## １．　はじめに

ICOTでは第５世代コンピュータの実現をめざして、知識ベース管理システム、問題解決・推論システム、知的インタフェースシステム、知的プログラミングシステムに関連する各種基礎ソフトウェアの研究開発（図１参照）を行なっている。なかでも知識ベース管理システム、数式処理システム、核言語第１版、知識プログラミングシステム、類推システムなどの論理プログラミング言語Prologを用いた研究試作において、従来の論理プログラミングの枠を越えるメタプログラミングの重要性[Kunifuji 85-4] が認識された。メタプログラミングは、与えられたプログラミング言語環境において、新たな拡張された言語機能をその言語自身で作り上げる機能であり、PrologのPrologによるユーザのための言語機能拡張機能である。

本論文で取り上げる話題は、論理プログラミング言語Prologを用いたメタプログラミング技法による知識獲得支援技術の研究開発の現状と今後の課題についてである。

まず第２章で、人間による問題解決・推論の記号論的本質を明らかにする。第３章と第４章で、ICOTで開発されたメタプログラミング方法論を要約する。すなわち第３章では、本論文で活用するメタ推論の基本的考え方を述べる。第４章では、メタプログラミング方法論の有効性を性能面からも保証する部分計算によるメタ推論の高速化技法について述べる。ついで第５章から第７章では、知識ベース管理の基礎技術として研究開発されてきた知識獲得支援機構の実現例を述べる。すなわち、第５章で知識同化機構の実現例を、第６章で知識調節機構の実現例を、第７章でトランザクション管理機構の実現例を紹介する。さらに第８章と

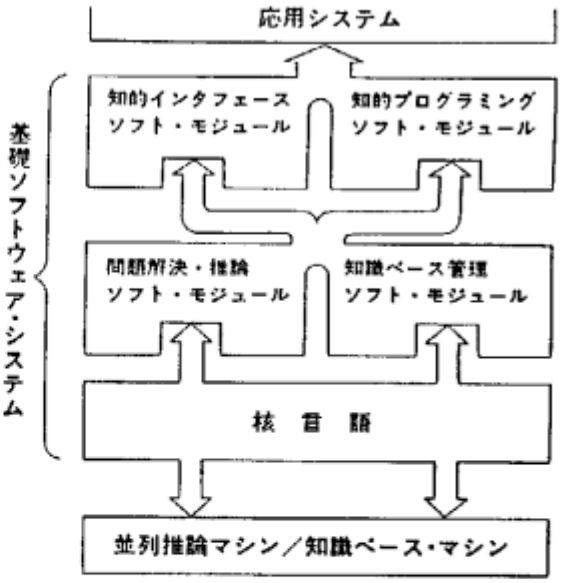


図１　基礎ソフトウェア・システム

第９章では、問題解決・推論の基礎技術として研究開発中の知識獲得支援機構の実現例を述べる。すなわち、第８章で仮説生成・検定機構の実現例を、第９章で類推機構の実現例を紹介する。最後に第１０章では、知識獲得支援のための基礎技術確立をめざして、今後の課題を要約する。

## ２．　問題解決と推論

### ２．１　問題解決・推論のプロセス

人間の問題解決のプロセスを、最も緻密な論理学の体系の中で、初めてふれたのはアリストテレスの分析論である。彼はその中で問題解決の論理的方法として、演繹法(deduction)・帰納法(induction)・発想法(abduction)という３つの推論プロセスを挙げている。そのうち演繹法の体系は、その後中世に至るまで、学問の明晰な方法論として、順調すぎるほど順調に成長していった。他の２つの方法論は、長い間、まどろみの時代を過ごした。近世になってベーコンやミルが帰納法の体系の再発見を行い、20世紀に入って推測統計学や科学哲学といった学問の世界で、帰納論理の急速な整備が行われた。一人取り残されていた発想法の体系化については、19世紀後半から20世紀初頭にかけて生存した独創的な哲学者パースの活躍を待たねばならなかった。

### ２．２　パースの記号学

19世紀後半から20世紀初頭にかけて幾多の著作を残したプラグマティストC.S.パースは、先駆的な記号学者として人間の探求活動の全過程を演繹論理、帰納論理、発想論理[Yonemori 81]の立場から体系化した。

彼の記号学によれば、人間の探求過程とはある未知の問題（驚くべき事実）に遭遇した人間が、その問題を説明する仮説を発見し（発想）、その仮説から導き出される合理的な帰結を推論し（演繹）、そして導出された帰結を検証する（帰納）過程である。

そこで人間の知的問題解決活動を支援する知識ベース管理システムを構築するため、演繹・帰納・発想という探求過程を支援するシステムを作るということを最終目標とする。

### ２．３　演繹・帰納・発想の機械化

上述のような観点からすると、演繹・帰納・発想[Kunifuji 81,83-1,85-3,86,Arikawa 85,86]の既存計算機システム上での実現方式に注目したい。既存の計算機システム上で演繹的あるいは帰納的な推論機構を実現するには、限られた論理の世界を前提とすれば、分解証明法(resolution principle )あるいはモデル推論法(model inferencemethod)というアルゴリズムを用いればよいことが知られている。[Kunifuji 83-1, Kitakami 84-1,84

-2] 。発想的推論機構実現のための統一原理は、現時点ではほとんど知られていないが、その中核をなす仮説生成(hypothesis generation) や類推(analogical reasoning)については、最近、Poole ら[Poole 85]や原口[Haraguchi 84.85-2] がそれらの基礎理論を構築中であり、その機械化のための突破口がようやく見えてきたところである。

## 3． メタ推論

### 3．1 メタ知識

論理型言語Prologで処理可能な知識は、基本的に一階述語論理の部分クラスであるホーン節である。ホーン節は論理的には帰結部の不確かさを含まない知識、すなわち結論が高々一つしかない知識、の知識表現に適している。このようなホーン節を知識表現の礎とするとき、人間のもつ多様かつ不確かな知識をも処理対象とするには、論理型言語としてのProlog（いわゆるpure Prolog ）の諸機能でなくプログラミング言語としてのPrologの諸機能（具体的には論理の枠を越える組込み述語）を用いて、そのような知識を実証（demonstrate)あるいは模倣（simulate）する機能を実現してやればよい。

Prologでは、現実世界の対象に関する知識はファクト（事実）またはルール（規則）として取り扱われる。ここではファクト型知識やルール型知識の容物を知識ベースと呼ぶことにする。しかしながら、現実世界にはファクトやルールといった対象世界に関する知識以外に、そのような対象知識の使い方に関する莫大なノウハウ型知識や発見的知識がある。このような知識をメタ知識と呼ぶことにすれば、メタ知識は対象知識の使い方に関する知識、あるいはメタ知識の使い方に関する知識として再帰的に定義される。

さてメタ推論とは、メタ知識を用いた推論のことである。メタ知識の基底をなす対象知識としては、論理型言語Prologで表現可能なホーン論理の節集合が利用される。メタ推論機構の提供とは、具体的にはPrologインタプリタ／コンパイラの使い方に関する知識表現・知識利用技術を確立することにある。 3.2と 3.3でメタ推論の実現法を与えることにする。

### 3．2 demo述語によるメタ推論

ICOTでは知識情報処理指向の各種アプリケーションを研究開発するために、論理プログラミング言語Prologを土台とする核言語ファミリィを提供しつつある。逐次型（並列型）推論マシン上の機械語がKL0 （KL1)で、そこにおけるメタ推論用プリミティブがdemo（simulate）述語といわれる。demo述語とsimulate述語は概念的には等価であり、その相違は主として逐次型計算環境と並列型計算環境という計算環境の相違である。

ここではまず、逐次型Prologでのメタ推論方式について要約する。メタ推論用demo述語としては、次のような形式のものが最も一般的であり、ICOT[Kunifuji 83-3] によって提

案された。

① demo(World, Goal, Result, Control)

このdemo述語は、ある世界World において、与えられた制御情報Control に従って、与えられたゴールGoalを証明していくプロセスを実際に示し、その証明のプロセスから必要なメタ情報Resultを抽出する。ここにControl とは証明プロセスを制御するために与えられた戦略情報であり、Resultとは証明プロセスの履歴から抽出される与えられたゴールを解決するのに必要な情報を保持していく引数である。

上述のdemo述語①に対して、次のような省略した形式のdemo述語がしばしばある種の応用では有効である。その理由は、主としてそれらの実現の容易さと性能の向上のためである。

② demo(World, Goal, Result)

③ demo(World, Goal)

なお対象世界として、Prologの内部データベース自身を用いる場合、①～③の第１引数World は省略できる。たとえば、次のメタ述語④は"Prolog interpreter in Prolog " あるいはPrologのメタインタプリタとして知られている。

④ demo(Goal)

## ３．３ simulate述語によるメタ推論

次に並列型Prologでのメタ推論方式について要約する。並列メタ推論方式として提案されている最も一般的なメタ推論用simulate述語は、次のような形式[Kunifuji 83-3] をしている。

⑤ simulate(World, NewWorld, Goal, Result, Control)

このsimulate述語は、ある世界World において、与えられたゴールGoalを与えられた制御情報Control 下で解くプロセスを模倣する。模倣の結果、世界World は新世界NewWorldへ更新されると同時に、ゴールを証明していくプロセスそのものから必要なメタ情報Resultが順次抽出されていく。

上記のsimulate述語⑤に対して、次のような省略した形式のものが、しばしばある種の応用では有効である。

⑥ simulate(World, NewWorld, Goal, Result)

⑦ simulate(World, NewWorld, Goal)

⑧ simulate(World, Goal)

なお、対象世界World として、内部データベース自身を用いる場合は⑥～⑧の第一引数 World は省略できる。たとえば、次のメタ述語⑨は"KL1 interpreter in KL1"あるいはKL 1 のメタインタプリタとして知られている。

⑨ simulate(Goal)

## 4． 部分計算によるメタ推論の高速化

### 4．1 部分計算

メタ推論を用いたメタプログラミング技法は数式処理における推論の制御[Takewaki 85 -1,85-2]、知識プログラミングにおけるユーザ定義推論エンジンの導入、知識ベース管理における知識同化や知識調節[Kunifuji 85-1,Kitakami 85] といった要素技術の実現等において極めて有効であった。しかも並列型Prologへのメタプログラミングの適用例も、最近急激に増えつつある。

メタプログラミング技法の唯一の欠点は、インタープリティブに走る所が多いので、処理性能が遅いことである。しかしながら最近ICOTでは、部分計算によるメタプログラミング技法の高速化技術を確立した。本章では、その成果を要約する。

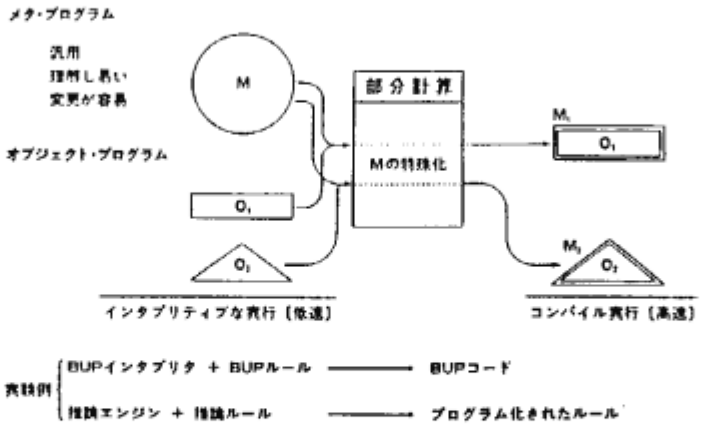


図2 メタプログラムの高速化

プログラムの部分計算とは一般に『稼働環境に関する情報を利用して、汎用のプログラムをより効率の良いプログラムに特殊化すること』である。部分計算の原理は任意のプログラミング言語で書かれたプログラムに適用でき、応用上の意義もコンパイラの自動作成など非常に大きい。部分計算の原則は与えられたプログラムの中で、計算できる部分を計算し、計算できないものは元のままに残して置くことといえる。一般に関数型言語のような値指向の計算においては、変数の値の有無が計算できる、できないにつながり、値がないままに計算する場合には遅延評価などの特殊な評価方式をとらなくてはならない。しかし、Prologの計算は、ユニフィケーションをベースにしているので、基本的に部分計算時に特殊な評価方式を必要としない。このことはPrologでの部分計算の重要な特徴である。

## ４．２　推論エンジンの高速化

Prologプログラムの部分計算法PEVAL[Takeuchi 85-1,85-2] はメタインタプリタ的なプログラムを効率の良いプログラムへと変換するのに極めて有効である。実際、図２に示されているように、メタインタプリタにとってオブジェクト・プログラムは入力データ的なものであるので、メタプログラムにオブジェクト・プログラムを与えて部分計算し、特殊化することができる。この特殊化されたプログラムは、機能的にはメタインタプリタ的なものを用いずに、すべてをオブジェクト・レベルのプログラムに埋め込んだものに極めて近くなる。従って効率についても、特殊化されたメタプログラムは、メタインタプリタを使わなかったものと同等になる。

| | p1 | p2 | p3 | p4 |
|---|---|---|---|---|
| インタプリティブ実行 | 1674 | 901 | 1157 | 95 |
| コンパイル実　行 | 110 | 39 | 46 | 26 |

(CPU Time)

表１　実行時間比較[Takeuchi 85-2]

試作されたPEVAL の効率の向上について述べる。ここにp1は、確信度(CF)を扱う最も簡単なプロダクション・システムでPrologメタインタプリタとその解釈実行されるオブジェクト・レベルのプログラムとからなる。p2は部分計算の前向き進行制御を行うtype述語をデータとして付与した場合の、部分計算の結果であり、この問題向きの特殊化されたプログラムとなっている。p3はさらに、部分計算の後向き進行制御を行なう inhibit_unfolding 述語を付与した場合の部分計算の結果であり、p2よりオブジェクト・プログラムに良く似た構造をした特殊化されたメタプログラムとなっている。p4はCFなしの場合の結果で

ある。以上述べたp1～p4プログラムの実行効率の比較を表１に示す。表１より、コンパイル実行ではp2プログラムの効率がp1プログラムの約 3倍であり、またp4プログラム単体の実行効率の約2/3 であることが分る。なおICOTで試作した数式処理システムをPEVAL で部分計算した結果[Takewaki 85-1,85-2]、人間がハンド・コンパイリングした理想的な状態より、9%程度の効率の損失で済むことが分かった。このことはメタプログラミングの解り易さ、開発のし易さからいっても、本手法の有効性を実証している。

メタプログラミングはPrologプログラミングの中でその表現力の強力さ故に重要な役割を果しつつあるが、部分計算はこのメタプログラムの実行効率を向上させることができ、このメタプログラミングをより実用的なプログラミング技法にすることができる。

## ５．　知識同化機構の実現

### ５．１　知識ベース管理と知識獲得

ここではメタ推論による知識獲得機能の実現例として知識ベース管理の実現を取り上げる。ICOTでは、知識獲得機能を中核とする知識ベース管理システムを、論理プログラミング言語Prologを用いて研究試作中[Kunifuji 85-1, Kitakami 85]である。そのようなシステムの研究試作の途上で、次のような知識ベース管理に関する要素技術が明らかになった。

(1) 与えられた知識ベースが正しいと仮定した時、外から与えられた知識をその知識ベースに無矛盾かつ系統的に取り込むという過程の管理、すなわち知識同化（knowledge assimilation）[Kunifuji 83-3, Miyachi 84-1,84-2]という知識ベース管理の基本技術を明らかにした。

(2) その際、論理型言語Prologでの無矛盾性管理の仕方を明らかにした。これは、統合性制約[Kunifuji 83-3, Miyachi 84-1,84-2]に基づく論理データベース管理の考え方の自然な拡張となっている。

(3) 外から与えられた知識が正しいと仮定した時、そのような知識を論証しうるモデルを構築するように、知識ベースそのものを無矛盾かつ系統的に修正していくという過程の管理、すなわち知識調節（knowledge accommodation)[Kitakami 84-1,84-2, Kunifuji 85-2] という知識ベース管理の基本技術を明らかにした。

(4) 知識同化や知識調節という要素技術の抽出を通じて、論理型言語での知識獲得のパラダイム[Kitakami 84-2,84-3, Kunifuji 85-2] を提案した。これは従来から人工知能で言われている学習のパラダイムの統合であり、演繹的知識獲得と帰納的知識獲得の自然な統合である。

(5) その際、demo述語によるメタ推論の方式を提案し、多くの適用事例[Kunifuji 85-2, Kitakami 84-3]を与えた。これにより論理型言語におけるメタプログラミング技法の有用性を実証した。

## ５．２　ファクト型知識同化機構の実現

知識同化機構のメタ推論による実現例として、ファクト型知識同化機構の実現法を示す。与えられた知識ベースCurrkbに外から取り込みたい新知識Input を入力し、その結果、CurrkbをNewdb へと更新する知識同化プログラム"assimilate(Currkb,Input,Newkb)"は、次のような抽象的プログラムとして実現[Kunifuji 83-2, Miyachi 84-1] できる。

```
assimilate(Currkb,Input,Currkb):-demo(Currkb,Input)
assimilate(Currkb,Input,Currkb):-demo(Currkb∪Input,False).
assimilate(Currkb,Input,Newkb):-Info∈Currkb,Interkb=Currkb-Info,
                                demo(Interkb∪Input,Info),
                                assimilate(Interkb,Input,Newkb).
assimilate(Currkb,Input,Currkb∪Input):-independent(Currkb,Input).
```

ここに∪、－、∈は集合論の通常の記法である和集合、差集合、メンバシップの意味である。本述語は、知識ベースに新知識を同化するには、証明可能性、矛盾性、冗長性、独立性という四つの概念のこの順序での検査と対応する知識ベース更新の必要性を示している。本例に典型的に見られるように知識ベース管理機能は、基本的にdemo述語を利用したメタ推論方式を用いて達成される。

## ６．　知識調節機構の実現

### ６．１　知識調節とモデル推論

知識調節機構において、外から与えられた知識を論証（説明）するモデルを知識ベース内に構築するのに、帰納推論の一種であるモデル推論アルゴリズム（model inference algorithm ）を用いる。Shapiro によって提案されたモデル推論アルゴリズムは、図３に示されているように、ファクト型知識（経験的知識）からルール型知識（一般的知識）を帰納的に推論する学習アルゴリズムである。その際、ユーザは誤りのないファクトを与えることが要請されるので、彼のシステムは完全教師付きの単一概念の学習システムといえる。

Shapiro は図３で与えられるモデル推論アルゴリズムをホーン論理に適用できるように洗練し、PrologのPrologによるPrologのための知的デバッガを開発した。彼の知的デバッガはモデル推論システムとも呼ばれるが、その中核をなすモデル推論アルゴリズムは矛盾を発見するプログラムと反証を与えた仮説を精密化するプログラムからなる。これら両プログラムは前述のdemo述語を用いてもインプリメントできる[Kitakami 84-1,84-3]。

### 6．2　知識同化・調節と知識獲得プロセス

ICOTでは、前述の知識同化機構と上述の知識調節機構の両者を中核とし、Prologによる知識獲得支援システムを構築した。知識獲得支援システムにおいては、知識同化・調節機構を用いて、知識ベースへの知識の獲得の管理を行なっている。

知識同化と知識調節という考え方を統合する際、最も注意を払ったのは図4に見られるように、帰納推論によって得られた論理プログラム（仮説）が正しいかどうかをユーザによって確認する作業である。現在はこの仮説確認実験をユーザの判断に基づいて行なっている。この部分は、概念的には、仮説の検定を行なっていることを留意しておきたい。

```
Tを {□} と設定する.
repeat
  つぎの事実を読む.
  repeat
    while  推測Tが強すぎる do
      矛盾後戻りアルゴリズムを適用し，Tから反
      証を与える仮説を取り除く.
    while  推測Tが弱すぎる do
      先に反証を与えた仮説を洗練したものを，T
      に加える.
  until  推測Tが強すぎもせず弱すぎもせず.
      (読み込まれた事実に関する限り)
forever.
```

図3　モデル推論アルゴリズム

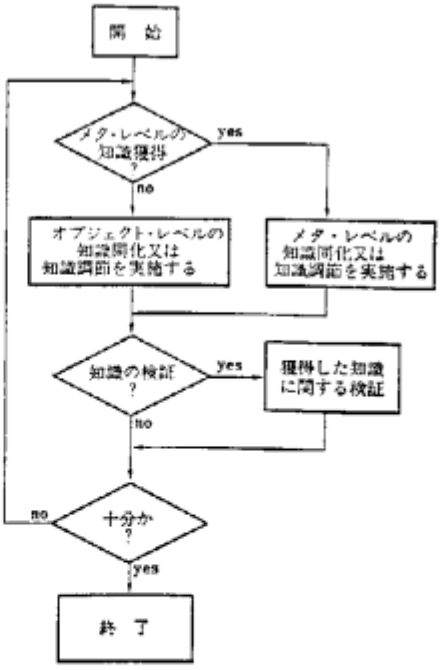


図4　知識獲得プロセス

## 7．　トランザクション管理機構の実現

### 7．1　知識ベース管理機構の拡張

第5章や第6章で述べた知識ベース管理の基礎技術は、更に次のような知識ベース管理のための基礎技術[Kitakami 85]を明らかにする引き金となっ

た。

(1) 構造化知識としての概念の階層関係を簡潔な形で表現し、この関係の関係に関する（本来は高階述語で表現される）諸性質を系統的にルール化し、それをメタプログラミングによって実現した。

(2) メタ推論を利用して制約用メタ知識を容易に評価実行できることを実証した。なかでも、自然言語処理で良く使われる因果関係の表現に、知識ベースの更新オペレーションを管理するメタ知識であるtrigger 述語が有効に適用できることを示した。

(3) 制約用知識の評価実行のタイミングを明確にするために、即時型(immediate-type)メタ知識と遅延型(delayed-type)メタ知識という評価実行のタイミングを表わす概念を取り入れた。特に遅延型メタ知識の実行管理を行う場合、トランザクションという処理単位で統合することの重要性を具体例をあげて実証した。

(4) メタレベルの知識を同化する場合、オブジェクト・レベルの知識のうち信念と呼ばれるものの調節を実現するための基礎技術として、矛盾知識の抽出法とその効率的な修正法を明確にした。

## 7．2　トランザクション管理

上述の知識ベース管理のための拡張機構のうち、知識獲得支援機構として必須なのはトランザクション管理[Kitakami 85] である。トランザクション管理には、メタ知識の一種である制約型知識を与える必要がある。制約型知識には、次の種類がある。一つは、知識ベースの更新アクセスなどのオペレーションに対し、動的に整合性をとるために、種々の知識の追加または削除を行う機能をもつtrigger 型のメタ知識である。他の一つは、知識ベースの更新アクセスとしてのオペレーションに対する無矛盾性を監視するinconsistency 型のメタ知識である。このメタ知識は、オブジェクト知識が矛盾する条件を記述しているので、否定型ルールと見なすことができる。両メタ知識ともに、トランザクション処理の最後に起動される遅延型のメタ知識と、トランザクション処理とは独立であって更新などの知識アクセスがあることに起動される即時型のメタ知識とに分類される。

知識ベースに対する更新は、知識獲得プロセスの一環として、実施されるが、この更新により、知識が誤った方向に成長することを防止するための判定が制約型知識により常に監視されている。この判定のタイミングには、更新があるたびに行われる場合と、ある一連の更新群が終了した後に行われる場合の二つがある。後者は、一連の更新群を定義するためにトランザクションという処理単位が必要であり、このトランザクション処理の中では、遅延型の制約型知識の扱いが重要になってくる。

トランザクション処理の重要性を示すために、トランザクション処理の例を図５に示す。図中の最初の例は、is_a(chickweed, vegetable) というis_a 関係の知識挿入例である

が、「Yの上位概念が生物でないようなis_a 関係"is _a(X,Y)" を生物というフレーム"frame" に追加するなどということが、あってはいけません」というメタ知識に違反したので、この処理が不成功に終っている。ところが、次の例では、is_a(chickweed,vegetable)及びis_a(vegetable, plant_for _food)を同じトランザクション処理内で知識を挿入しているので、この処理が成功している。

```
| ?- super_concept(frame, X, plant).
X = plant_for_food;
X = fruit;
no
| ?- transaction(insert(frame, is_a(chickweed, vegetable))).
* Transaction Error.
yes
| ?- transaction((insert(frame, is_a(chickweed, vegetable)),
|                 insert(frame, is_a(vegetable, plant_for_food)))).
* End Transaction.
yes
| ?- super_concept(frame, X, plant).
X = plant_for_food;
X = fruit;
X = chickweed;
X = vegetable;
no
```

図5　トランザクション処理の例

## 8.　仮説生成・検定機構の実現例

### 8．1　発想と仮説生成

第8章と第9章では、発想推論を利用した問題解決・推論の基礎技術として研究開発中の知識獲得支援機構について述べる。さて発想とはある驚くべき事実（困難な未知の問題）に遭遇した人間が、その事実を説明する（その問題を解決する）仮説を直観的に得る人間の思考過程である。前節で述べた帰納的推論は、ある閉じた知識体系の中で極限としての同定[Arikawa 83]をめざした仮説の生成を行なっている。しかしながら、本当に発想的推論と呼ばれうるものでは、その知識体系に関する知識（ファクトとルール）が不十分なので、人間は種々の知的技法を用いて、何らかの意味のある仮説を生成している。意味のある仮説生成の仕方により、各種の発想的推論のパターンが分類される。第8章（第9章）では、そのうち仮説生成（類推）と呼ばれるものだけを述べる。

### 8．2　仮説生成・検定

一般に仮説生成（hypothesis generation ）とは、経験的知識である結論（観測された事実"B" ）と一般的知識である大前提（既知の規則"A⊃B" ）とから、個別的知識である小前提（未知の仮説"A" ）を推測する推論過程である。すなわち、仮説生成を一階述語論理の用語法を用いて直観的に説明すると、次のような推論図式となる。

$$\frac{B \qquad A \supset B}{\therefore A}$$

この推論図式を科学的理論の理論形成（theory formation）システムへ展開したのが、Poole らのTheorist[Poole 85]である。Theoristでは図６に見られるように、与えられた観測事実の集合を説明する適切な理論を、作り付けの可能な仮説集合の中から選択していく。ただし、観測された事実を説明するのに既知の事実の集合（知識ベース）のみからは説明できず、それと無矛盾な適切な理論を選んで初めて与えられた観測事実が説明できるものとする。Theorist研究に刺激を受けて、鶴巻ら[Tsurumaki 85]は図７に見られるような構造化された知識の世界での仮説の生成・選定システムを構築中である。本システムは仮説の生成・選定機構を内蔵しており、仮説生成・選定の評価基準を明確に実験しうる“思考のワークベンチ”として今後の発展が期待できる。なお両システムともメタプログラミング技法を活用してインプリメントされている。

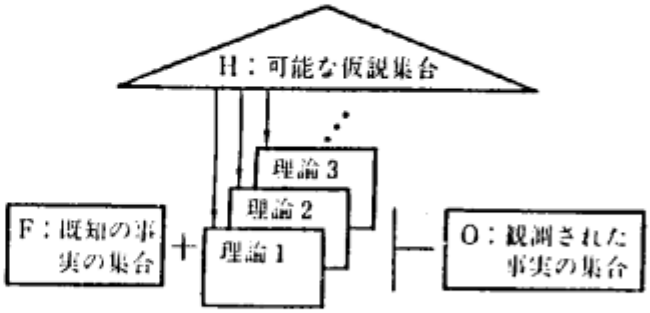


図６　Theoristのフレームワーク

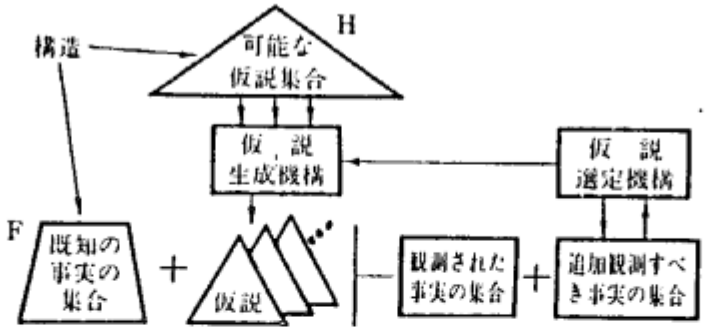


図７　仮説生成・選定システムのフレームワーク

## ９．　類推機構の実現例

### ９．１　類推

一般に類推（analogical reasoning）においては、他の既知の知識体系との間に何らか

の翻訳・逆翻訳（創造工学の用語法では等価変換と呼ばれる）を行ないつつ、その事実を説明する仮説を生成する。類推の推論図式は、上述の仮説生成の推論図式に対する観点を変えることによって得られる。すなわち、図８に見られるように、未知の知識体系においてある驚くべき事実Ｂ´が観測されたとする。このＢ´に対して、既知の知識体系においてＢ´と類似の事実Ｂが存在することを見出し、しかもその体系では“ＡならばＢ”が成立すると仮定する。すると類推とは、未知の知識体系においてＡと類似の事実Ａ´が存在し、しかも”Ａ´ならばＢ´”が成立するという推測をたてる根拠があるという推論図式のことである。Ｂ´とＢ（あるいはＡとＡ´）とを結び付けるには、一般に種々の等価変換理論的な写像関係の当てはめが必要である。この当てはめとは、一般的にいえば、未知の知識体系の既知の知識体系への写像関係の発見にほかならない。この写像を見出すのに、部分同型に基づく類比による思考形式が利用されているとするのが、原口の類比の理論[Haraguchi 84]である。この類比の理論をファクト型知識、ルール型知識の両者に対して適用したのが原口の類推の理論[Haraguchi 85-1,85-2] である。ルール型知識との類推においては、前向き・後向きの双方向の推論プロセスが存在するのが興味深い。この理論の基底をなす所与の部分同型そのものを、どのようにして人間が見出しているかについては、現段階ではまったく未知の問題領域である。おそらく、連想に基づく知的なパターン照合機構などが活用されているのであろう。

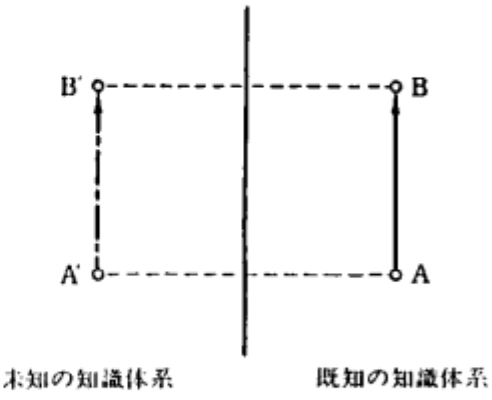


図８　類推の推論図式

## ９．２　類推

メタプログラミング技法を用いると、演繹・機能のみならず類推を用いた知識獲得支援機構を実現[Tshurumaki 85] することができる。ICOTで研究試作中の類推システムは、原口の類推理論[Haraguchi 85-1]にその基盤を置いている。従って、その中核となる部分は彼の類推システム[Haraguchi 85-2]と全く同じであり、次のようなメタプログラムで与えられる。

```
analogy(W,true):-!.
analogy(W,(Goal1,Goal2)):-analogy(W,Goal1), analogy(W, Goal2).
analogy(W,Goal):- wclause(W,Goal,Body),analogy(W,Body).
analogy(W,Goal):- prematch(W,Goal,TW,Tgoal,Tbody),
```

```
                    transform(W,Goal, Body,TW,Tgoal,Tbody,Apair),
                    analogy(TW,Tbody),analogy(W,Body),call(Apair),
                    message(6,['Transformation has succeeded :']),
                    message(10,['resulting analogy is',Apair]).
analogy(W,Goal):- message(0,['***Goals',Goal, 'in' ,W,'fails***']),fail.
```

上記類推機構の実現法を土台にし、次のような拡張機能の付与を検討中である。

(1) ファクト型ルールのみならずPrologの一般的なルールをも取り扱えないと、類推機能が貧弱である。従って、そのようなルールをも類推に利用する機能の付与が必要である。特に、閉世界仮説での否定の導入[Haraguchi 86]が本質的である。
(2) 述語の引数間の類推のみならず述語名間の類推をも発見する機能の付与が、一般の問題解決には必須である。
(3) ある種の幾何の定理証明が類推により解けるようになるには、予測推定型類推のみならず問題解決型類推[Arikawa 85,86] の機能の付与が必要である。

いずれにせよ、メタプログラミング技法を利用すれば、類推をベースとする発想の本質を議論する“思考のワークベンチ”を提供することができる。

## １０．　おわりに

論理プログラミングにより知識獲得支援機能をもつ知識情報処理システムのプロトタイプを構築するという一大目標を達成するには、解決すべき多くの課題をかかえている。この一大目標を攻略する一つの方法論として、メタプログラミングによるアプローチについて論じた。論理プログラミングにおけるメタプログラミング・アプローチは、理論的にも技術的にも、現在急激にその方法論が整備されつつある。

ICOTでのアプローチは、与えられた知識情報処理の各種基本問題に対して、まず論理プログラミングに対してdemo述語やsimulate述語に基づくメタ推論方式を確立し、さらにユーザにとって満足する性能を提供するために部分計算によるメタ推論の高速化方略を確立してゆくという二段階の手順を経た。今後、メタプログラミング技法はその実用性がますます増大することと思われる。

ついでメタプログラミングを用いた知識ベース管理のための基礎技術として、知識同化、知識調節、トランザクション管理機構の実現例を述べた。また問題解決・推論のための基礎技術として、仮説生成・検定機構、類推機構の実現例も述べた。以上を通して一貫して強調してきたことは、メタプログラミングによる演繹・帰納・発想的推論の機構化研究との関連である。メタプログラミングにより演繹・帰納・発想的推論機構を持つ知識情報処

理システム構築の道筋が見えてきたことも強調した。

本論文で述べたICOTの研究は、論理プログラミングによる知識獲得支援システム構築の第一歩を与えるものである。ICOTの研究はいまだ実験段階にあるとはいえ、論理プログラミングと知識獲得支援技術との間に横たわる深くて広いギャップを埋め、両者の橋渡しをとる一本の道を見い出している。この道は、方法論的にはメタプログラミングによる論理プログラミングと知識獲得支援技術の融合ということである。

〔謝辞〕　本稿をまとめるにあたって、日頃ご指導いただく古川康一室長（ICOT）、共同研究者として前期3年間お世話になった北上　始（現（株）富士通研究所）、宮地泰造（現三菱（株）情報電子研究所）の両氏、中期の共同研究者として日頃ご討論いただく竹内彰一、武脇敏晃、世木博久（ICOT）、鶴巻宏治（NTT 複合通信研究所）氏をはじめとする古川研究室の諸氏に感謝する。

## 〔参考文献〕

[Arikawa 85]有川節夫：帰納推論と類推、日本創造学会編、創造性研究第3巻“創造と企業”、共立出版、1985.

[Arikawa 86]有川節夫：帰納推論と類推－理論と応用－、渕　一博監修知識情報処理シリーズ第2巻“知識の学習メカニズム”、共立出版、1985.

[Haraguchi 84] Haraguchi,M.:An Analogy as a Partial Identity, Proc. of the Logic Programming Conference '84, 11-2, ICOT, March 1984.

[Haratuchi 85-1] Haraguchi, M. and Arikawa, S.: Analogical Reasoning based on the Theory of Analogy, Res. Rept. Res. Inst. Fund. Inform. Sci., Kyushu Univ., No.105, 1985.

[Haraguchi 85-2]原口　誠：ルールの変換による類推の逆向き推論について、Proc. of the Logic Programming Conference '85, ICOT, July 1985.

[Haraguchi 86] Haraguchi, M. and Arikawa, S.: A Foundation of Analogical Reasoning: Analogical Unions of Logic Programs, Res. Rept. Res. Inst. Fund. Inform. Sci., Kyushu Univ., No.111,April 1986.

[Kitakami 84-1] Kitakami, H., Kunifuji, S., Miyachi, T., and Furukawa, K.: A Methodology for Implementation of a Knowledge Acquisition System, Proc. of the 1984 International Symposium on Logic Programming, Atlantic City, U.S.A., Feb. 6-9, 1984.

[Kitakami 84-2] 北上　始、國藤　進、宮地泰造、古川康一：大規模な知識ベース管理システムのアーキテクチャ、知識理解システム・夏期シンポジウム、富士通㈱国際情報社

会科学研究所、昭和58年 7月.

[Kitakami 84-3] Kitakami, H., Kunifuji, S., Miyachi, T., Furukawa, K., Takeuchi, A., Miyazaki, T., Ishii, S., Takewaki, T. and Ohki, M.: Demonstration of the KAISER System at the ICOT Open Houce in FGCS'84, ICOT TR-0018, 1984.

[Kitakami 85] 北上　始、國藤　進、宮地泰造、古川康一：論理型プログラミング言語Prologによる知識ベース管理システム、情報処理、11月、1985.

[Kunifuji 81] 國藤　進：知識情報処理システムから創造科学へ、オペレーションズ・リサーチ、pp.261-268、1981年 5月号.

[Kunifji 83-1]國藤　進：問題解決と推論、計測と制御、Vol.22、No.1、昭和58年 1月、pp.153-159、1983.

[Kunifuji 83-2] 國藤　進、麻生盛敏、竹内彰一、坂井　公、宮地泰造、北上　始、横田治夫、安川秀樹、古川康一：Prologによる対象知識とメタ知識の融合とその応用、情報処理学会知識工学と人工知能研究会 30-1 、1983.

[Kunifuji 83-3] 國藤　進、竹内彰一、古川康一、上田和紀他：核言語第１版概念仕様書（案）、ICOT, 1983.

[Kunifuji 85-1] 國藤　進、北上　始、宮地泰造、古川康一：知識工学の基礎と応用［第４回］－Prologにおける知識ベース管理－、計測と制御、Vol.24、No.6、53-62. 1985.

[Kunifuji 85-2] 國藤　進、北上　始、宮地泰造、古川康一：論理型言語Prologによる知識ベースの管理、Proc. of the Logic Programming Conferente '85, ICOT, 1985.

[Kunifuji 85-3] 國藤　進：演繹・帰納・発想の推論機構化をめざして、日本創造学会編、創造性研究第３巻“創造と企業”、共立出版、1985.

[Kunifuji 85-4] 國藤　進、武脇敏晃、世木博久、竹内彰一、大木　優、古川康一、鵜巻宏治：メタプログラミングによる論理プログラミングと知識情報処理の融合、第28回自動制御連合講演会前刷“特別講演”、国立教育会館、pp. Ⅲ-1～Ⅲ-10 、1985年11月 7日.

[Kunifuji 86] 國藤　進：演繹・帰納・発想の推論機構化をめざして、渕　一博監修知識情報処理シリーズ第２巻“知識の学習メカニズム”、共立出版、1986.

[Miyachi 84-1] Miyachi, T., Kunifuji, S., Kitakami, H., Furukawa, K., Takeuchi, A and Yokota, H., 1984: A Knowledge Assimilation Method for Logic Databases, Proc. of the 1984 International Symposium on Logic Programming, Atlantic City, U.S.A., Feb. 6-9, 1984.

[Miyachi 84-2] Miyachi,T., Kunifuji, S., Kitakami, H. and Furukawa, K.: A Knowledge Assimilation Method for Logic Databases,New Generation Computing, 2-4, 385/404, 1984.

[Miyachi 84-3]宮地泰造、國藤　進、古川康一、北上　始：Constraintに基づく論理デー

タベースの管理について、情報処理学会知識工学と人工知能研究会、36-8, 1984.
[Poole 85] Poole, D., Aleiunas, R. and Goebel, R.: Theorist: A Logical Reasoning System for Defaults and Diagnosis, in Knowledge Representation (Cercone, N.J. and McCalla, G. (eds.)), IEEE Press, in preparation.
[Takeuchi 85-1] 竹内彰一、近藤浩康、大木　優、古川康一：部分計算のメタプログラミングへの応用、情報処理学会ソフトウェア基礎論研究会、1985.
[Takeuchi 85-2] 竹内彰一、古川康一：Prologプログラムの部分計算とメタ・プログラムの特殊化への応用、Proc. of the Logic Programming Conference '85, ICOT, 1985.
[Takewaki 85-1] Takewaki, T., Miyachi, T., Kunifuji, S. and Furukawa, K.: An Algebraic Manipulation System using Meta-level Inference based on Human Heuristics, First International Expert Systems Conference, London, 3 October 1985.
[Takewaki 85-2] Takewaki, T., Takeuchi, A., Kunifuji, S. and Furukawa, K.: Application of Partial Evaluation to the Algebraic Manipulation System and its Evaluation, ICOT TR-148, Dec. 1985.
[Tsurumaki 85]鶴巻宏治、國藤　進、古川康一：メタプログラミングによる類推システムの試作について、日本ソフトウェア科学会第２回大会論文集、1985年11月.
[Yonemori 81] 米盛裕二：パースの記号学、勁草書房、1981.