ICOT Technical Report: TR-129

TR-129

メタプログラミングによる
知識情報処理への接近

國藤　進，竹内彰一，武脇敏晃
大木　優，古川康一

July, 1985

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情報処理への接近

渡部　透、竹内彰一、武脇敬晃、大木　優、古川康一
（財）新世代コンピュータ技術開発機構

1．はじめに

近年、人工知能や知識工学といった研究分野の急激な進展の刺激をうけ、知識の表現、知識の利用、知識の獲得に関する基礎・応用・開発研究の成果を集大成し、知識情報処理システムのプロトタイプを構築しようという動きがさかんである。知識情報処理システム構築の気運の胎動は、歴史的に見ると、学界や産業界における人工知能研究の見直しの動きあるいは物質と無縁ではない。周知のように人工知能研究の歴史は、1960年代前半まではゲームとパズルの時代、1960年代後半は知能ロボットの時代、1970年代は言語と知識の時代、そして1980年代は知識工学と認知科学の時代といわれている。

著者らの所属するICOTは、いわゆる第五世代コンピュータの研究開発を目途として設立された財団法人である。第五世代コンピュータは1990年代における知識情報処理システムのプロトタイプ実現をめざすコンピュータであり、その実現のためICOTでは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。

図1　基礎ソフトウェア・システム

知識情報処理システムのプロトタイプ実現にあたって、ICOTでは逐次型／並列型の論理プログラミング言語Prologを用いた各種の基礎ソフトウェアの実験的試作を行っている。なかでも核言語第1版、知識プログラミング言語、知識ベース管理システム、数式処理システム等の研究試作において、従来の論理プログラミングの枠を超えるメタプログラミングの重要性が認識された。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。

本稿で取り上げる話題は、逐次型／並列型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情報処理技術の融合についてである。第2章で人間による問題解決推論プロセスの本質を探り、第3章で知識情報処理への著者らのアプローチの特徴を述べる。第4章でメタ推論の基本的考え方、第5章でメタ推論による制御の実際、第6章でメタ推論に基づく知識ベース管理の仕方、第7章で部分計算によるメタ推論の高速化について述べる。

2．問題解決と推論

2.1　問題解決のプロセス

人間の問題解決プロセスの本質を、近い将来において機械において実行／模倣できるということを前提に、その本質を分析してみる。人間の知的問題解決活動を「問題提起、現状把握、本質追及、問題評価、決断、構想計画、具体策、計画評価、手順化、実施、検証、総括」といった一連のプロセスの中でとらえるのが、川喜田らの「問題解決学」である[1]。問題解決学は、異質のデータ・情報を統合することによって新しい発想とアイデアを産む方法論であるKJ発想法を中心とする知的生産の技術を駆使して、人間が毎日遭遇する問題を問題として意識して、自らの課題としてとりあげ、解決に向かっていく方法論を与えるものである。問題解決学を野外科学・実験科学・実践科学をつなぐプラグマチックな方法論（図2）とみるとき、前半のV（A～D部）を発想法のプロセス、真中のD～E部を演繹法のプロセス、後半のV（F～H部）を帰納法のプロセスとみなすことができる。

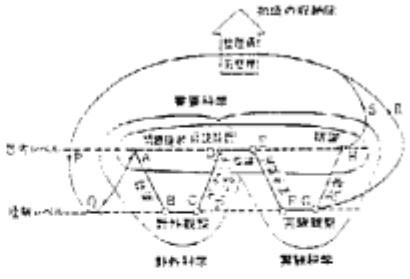

図2　W型問題解決学[1]

2.2　推論のプロセス

人間の問題解決のプロセスを、最も厳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・帰納法・発想法という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という学問の世界で帰納論理の急速な整備が行われた。一人取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。

2.3 パースの記号学

19世紀後半から20世紀初頭にかけて数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[2]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。この探求の3プロセスは、実はそれぞれ前述のW型問題解決学におけるA～D部分、D～E部分、E～H部分にほかならない。

以上から、人間の知的問題解決活動を支援する知識情報処理システムを構築するには、究極的には演繹・帰納・発想というプロセスを支援するシステムを作らなければならないことが分る。

3．知識情報処理への道

3.1 処理対象の拡大

情報処理装置としての計算機システムの近年の発展は、その処理対象をマシーンよりのものからヒューマンよりのものに急激に拡大しつつある。例えば初期の計算機システムは0、1という二進数の処理中心のものであったが、次第に数値処理やデータ処理向きの計算機システムが作られた。それに伴いユーザの使用する計算機言語も機械語、アセンブラ語、Fortran 、Cobol 、PL/1へと発展してきた。1970年代、1980年代になって新たに記号／知識処理向きの計算機システムを要求するニーズが高まり、それに伴いユーザもLispやPrologといった知識情報処理用の思考のツールとなりうるような使い易い処理系を求めつつある。

3.2 推論エンジンの変遷

このような計算機システムの知識情報処理システム志向への発展の中で、計算機システムのガソリンやエンジンにあたる部分も新たなる変容を遂げつつある。ガソリンにあたる部分は知識そのもので、エンジンにあたる部分は知識を処理する機構である。知識とそれを処理する機構の間には密接な関連がある。著者らの立場は、知識を容れる容物を、次のような三種のモデルに分類する。第一のものがファクト型個別知識を格納するデータベースと呼ばれるもので、第二のものがルール型一般知識を格納する知識ベースと呼ばれるもので、そして第三のものが個別知識や一般知識の使い方に関するノウハウ型知識を格納するメタ知識ベースと呼ばれるものである。これら三種のモデルに対応し、著者らはそれぞれ、データベースからの検索機構、知識ベースからの推論機構、メタ知識ベースからのメタ推論機構といった知識処理機構の必要性を喚起してきた。ソフトウェア的にみると、それぞれデータベース管理システム、知識ベース管理システムおよびメタ知識ベース管理システムとして実現できるものである。

3.3 アルゴリズム化の道

2章で述べたような観点から、計算機による知識処理の本質を分析するに、著者らは知的問題解決過程における人間の推論の本質である演繹、帰納、発想[3,4]の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、分解証明法あるいはモデル推論法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[3,4]。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、その中核をなす類推については最近、原口[5]が部分同型に基づく類推の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。

4．メタ推論

4.1 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。これ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ひとつしかない知識、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時、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（具体的にはメタロジカルな組込述語）を用いて、そのような知識を実証あるいは模倣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。

さて著者らの提案する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。次節でメタ推論の実現法を与えることにする。

4.2 demo述語によるメタ推論

第五世代コンピュータ・プロジェクト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。

ここでは説明の便宜上、逐次型Prologでのメタ推論方式についてのみ要約する。並列型Prologでのメタ推論方式は他の文献[6,17,18] を参照されたい。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、菅谷ら[6] 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。

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。例えば、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)

５．メタ推論による制御

5.1 数式処理の制御

一般に“論理プログラミング＝論理＋制御”ということが知られている。著者らは、Prologによる高校教科書や大学受験程度の問題を対象とした数式処理システム[7] を試作した。その際、人間のもっている数式解法のテクニックやノウハウをメタ知識として活用しないと、基本的に問題解決のための探索空間が広がりすぎることが分った。すなわち一般に、方程式を解くために使用可能な各種の規則を無制限に使用すると、探索空間が大きくなりすぎるため、探索空間を絞り込み、無駄な探索を押える必要がある。そこで著者らは、数式処理の制御にメタ推論を導入した。その特徴は、次の三つである。

(1) 複雑な処理制御の決定を容易に行なえる。

(2) 各数式処理規則のモジュール化が図れ、規則の追加変更が容易になる。

(3) 探索空間の絞り込みにより、無駄な探索を押える。

このメタ推論を実現するために、demo述語[7] を利用した。これにより、上記の特徴の他に、メタ推論部と、オブジェクト推論部が明確に分離し、拡張性に富むプログラムの実現が可能になった。

試作されたシステムは図３に示されているように、方程式の型の判定を行なう型制御部と適用可能規則の選択を行なう規則制御部の二種のメタ推論部をもつ。型制御部では、①代数方程式型、②無理方程式型、③分数方程式型、④指数方程式型、⑤対数方程式型、⑥三角方程式型、の６つの型の制御を行なっている。また規則制御部では①公式規則、②分離規則、③収集規則、④誘引規則、⑤置換規則、⑥変換規則、⑦因数分解規則、⑧展開規則の８つの規則の制御を行なう。

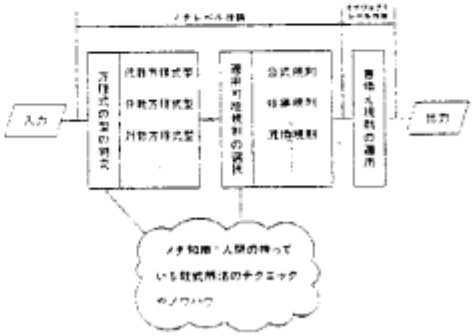

図３ メタプログラミングによる数式処理

5.2 規則制御の実際

demo述語による制御の実際を示すため、規則制御部の処理を行なうプログラムを示す。

```
method__demo( Eqn,Ctrl,Ans, _ ,[]):-
  end__check(Ctrl,Eqn,Ans),!.
method__demo( E1 && E2,Ctrl,Ans1 && Ans2,His,N3):-
  !,method__demo(E1,Ctrl,Ans1,His,N1),
  method__demo(E2,Ctrl,Ans2,His,N2),
  append(N1,N2,N3).
method__demo( Eqn,Ctrl,Ans,His,N__His):-
  strategy(Eqn,His,Rule , Ctrl,Ctrl1),
  apply(Rule, solve(Eqn,Ctrl1,Int, N__Ctrl, H1),
  _),append(H1,His,H2),
  method__demo(Int,N__Ctrl,Ans,H2,N3),
  append(N1,N3,N__His).
```

述語“method__demo”の引数は順に、方程式、制御情報、方程式の解、strategyのための規則使用履歴、トレースのための規則使用履歴を示す。第１節は、方程式Eqn が終了条件及び制約条件Ctrlを満す時に適用され、解をAns に返す。第２節は、方程式E1 && E2が分割可能を示す演算子&&で結合されている時に適用され、分割したそれぞれの方程式E1,E2 にmethod__demoを適用し、それぞれのトレースのための履歴N1,N2 をappendし、その値をN3に返す。第３節は、上記以外の時に適用され、strategyによって規則制御部の８種の規則の一つRuleとその時の制約条件Ctrl1 を抽

出し、それらに基づき、規則の適用をapply によって行なう。そしてapply の結果Int を再びmethod_demoに適用し、適用した規則の履歴H1をstrategyのための履歴His に追加しH2とする。そしてトレースのための履歴の追加登録を行なう。

数式処理システムの制御にメタ推論を導入することにより、複雑な数式の処理制御が容易に実現でき、数式処理の推論の過程の説明や問題の難易度を容易に付加することができる。解法規則として与えられた８種の規則と６種の方程式の型により処理を行なっているが、この枠組の決定は、処理効率の面からいっても、システムの最適な処理実行の構成上からいっても重要である。高次の代数方程式を解くための整係数の代数方程式に関する因数分解アルゴリズムなどを、有効に利用する方法の検討も重要である。数式を解く推論過程から得られた難易度の結果を学習し、これにより難易度が最小になるように規則適用順序を制御することも可能である。Prologによる数式処理において、メタ推論に基づく基本機能の容易な拡張性を確認できた。学習機能をもつ数式処理システムへの展開について、今後、検討していく予定である。

６．知識ベース管理

6.1　知識同化と知識ベース管理

若者らは知識ベース管理技術の中核をなす知識表現機能、知識利用機能、知識獲得機能をもつ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[8,9,10,11] である。そのようなシステムの研究試作の途上で、若者らは次のような知識ベース管理に関する要素技術を明らかにした。

(1) 与えらえ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込むという過程の管理、すなわち知識同化[12,13,14]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[12,13,14]に基づく論理データベースの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[15]という知識ベースの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[9] 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明らかにし、多くの適用事例[15,16] 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。

例えば、ファクト型知識同化機構は次のようにして実現される。Currkbを与えられた現存の知識ベース、Input をその知識ベースに取込みたい新知識とする時、知識同化の基本的考え方[11,12,13]は次のような抽象的プログラムとして実現できる。

```
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。

6.2　知識調節と知識ベース管理

一方、知識調節プログラムはShapiro によって提案されたモデル推論アルゴリズム（図６参照）を用いて達成された。Shapiro は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。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[9,15]。

```
Tを{□}と設定する．        {□：矛盾}
Repeat
  つぎの事実を読む．
  Repeat
    While 推測Tが強すぎる do
      矛盾探索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，さらに
      精密化したものを，Tに加える
  Until推測Tが強すぎることもなく弱すぎる
        こともない．
    （読み込まれた事実に関する限り）
  推測Tを答えとして出力する．
Forever
```

図４　モデル推論アルゴリズム

さて若者らは、前述の知識同化機構とShapiro のモデル推論システムに基づく知識調節機構の両者を中核にし、Prologによる知識獲得支援システムを構築中[11,16] である。知識獲得支援システムにおいては、知識の同化と調節の機構を用いて、知識ベースへの知識の獲得の管理を行っている。この両機構を併用

することにより、著者らは知識ベースへの知識獲得支援システム[15,16]へと統合中である。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帰納的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っている。この部分は、概念的には、仮説の検定を行っていることを留意しておきたい。

7．部分計算による推論エンジンの高速化

7.1 部分計算

前章までに述べたように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、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[17,18]、知識ベース管理における知識同化や知識調節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。メタプログラミング技法は知識ベースのエディタやデバッガの開発、論理プログラムの検証や自動合成、Prologと関係データベースとのインタフェース機構の実現、などにおいてもその有用性が確認されつつある。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。しかしながら最近著者らは、部分計算によるメタプログラミング技法の高速化技術[19,20]を確立した。本章では、その成果を要約する。

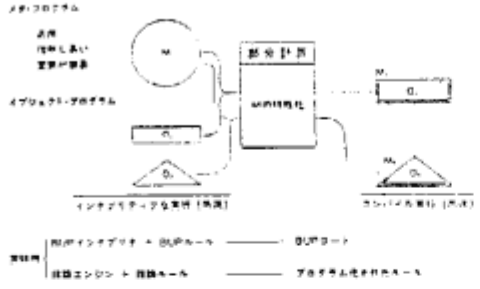

図5　メタプログラムの高速化

プログラムの部分計算とは一般に「稼動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」と定義される。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。部分計算の原理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言える。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。

7.2 推論エンジンの高速化

著者らの提案するPrologプログラムの部分計算法PEVAL[19]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5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のものになる。試作されたPEVALの効率の向上については、表1を参照されたい。

	p1	p2	p3	p4
インタプリティブ実行	1674	801	1157	65
コンパイル実行	110	39	46	28

p1：メタ・オブジェクト・プログラム
p2：特殊化されたプログラム
p3：特殊化されたプログラム
p4：certainty factorなしのオブジェクト・プログラム

表1　実行時間比較[19]

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、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。このことはPrologによる推論システム構築の新しい方法を示唆するものと考えることができる。また本手法はPrologの重要な特徴の一つである段階的プログラム開発の容易さとうまく調和でき、メタインタプリタの段階的特殊化の実現可能性の高いことも示した。このことは推論システムが段階的に推論ルールを獲得する場合においても、本方法が有効に働く可能性を示している。

今後の改良点としてはPEVALをより広い範囲のPrologプログラムを扱える強力なものにすること、およびプログラミング環境の中に部分計算をツールとして有機的に取り組む方法について研究することである。

8．おわりに

論理プログラミングにより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ひとつの方法論として、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。

著者らのアプローチは、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、ついでそれを用いた各種アプリケーションを開発し、更にユーザにとって満足するパフォーマンスを提供するために部分計算によるメタ推論の高速化戦略を確立してゆくという三段階の手順を経た。今後、メタプログラミング技法は知識の段階的コンパイル技法[20]と結び付き、その実用性がますます増大することと思われる。

メタプログラミングを論理型言語Prolog上で実現するに当って、メタ知識に基づくメタ推論機構実現の重要性を喚起してきた。メタ推論機構を用いた知識ベース管理機構の研究が着実に前進しつつあることも述べ、これが前述の演繹・帰納・発想的推論の機構化研究とも密接に関連しており、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を強調した。

以上述べてきたように著者らの研究は、論理プログラミングによる知識情報処理システム構築への第一歩を与えるものである。著者ら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情報処理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情報処理技術の融合ということである。

［参考文献］

1) 川喜田二郎、牧島信一編著：問題解決学ーKJ法ワークブック、講談社、1970.

2) 米盛裕二：パースの記号学、勁草書房、1981.

3) 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、261-268、1981年 5月号.

4) 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，創造性研究第３巻、1985.

5) 原口　誠：類推の機械化について、同上.

6) 國藤　進、竹内彰一、古川康一、上田和紀他：核言語第１版暫定仕様書（案）、ICOT (1983)

7) 武脇　敏晃、宮地　泰造、國藤　進、古川　康一：Prologによる数式処理システム試作の構想、日本ソフトウェア科学会第１回論文集，18-4，29-32，1984.

8) 古川康一、國藤　進、北上　始、宮地泰造：知識情報の取得と管理、昭和59年電気四学会連合大会32-3、1984.

9) 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・夏期シンポジウム報告書、富士通（株）国際情報社会科学研究所(1984)

10) 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.

11) 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 Conference '85，ICOT，1985.

12) 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)

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

14) 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.

15) H. Kitakami, S. Kunifuji, T. Miyachi, K. Furukawa : A Methodology for Implementation of A Knowledge Acquisition System, Proc. of the 1984 International Symposium on Logic Progrramming, Atlantic City, U.S.A.1, 131/142 (1984)

16) 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、情報処理学会誌、1985 （掲載予定）.

17) K. Furukawa, A. Takeuchi, S. Kunifuji, H. Yasukawa, M. Ohki, K. Ueda: Mandala : A Logic Based Knowledge Programming System, Proc. of the International Conference on FGCS'84 , 613-622, 1984.

18) 大木　優、古川康一、竹内彰一、國藤　進、安川秀樹、宮崎敏彦：Mandala Ⅱの拡張機能－ユーザ定義推論エンジンの実現方式－，日本ソフトウェア科学会第１回大会論文集1D-2，53-56，1984.

19) 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)

20) 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.