論理プログラミングと知識情報処理
ーメタプログラミングによるアプローチ

國藤　進，竹内彰一，北上　始，宮地泰造，大木　圀，武脇敏晃，古川康一

（財）新世代コンピュータ技術開発機構

東京都港区三田１－４－２８　三田国際ビル２１Ｆ(TEL 03-456-3193)

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。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を行なっている。

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の各種パラダイムの融合についてである。第２章でメタ推論の基本的考え方、第３章でメタ推論による制御の実例、第４章でメタ推論による言語機能拡張機能の実現、第５章でメタ推論に基づく知識ベース管理の仕方、第６章で部分計算によるメタ推論の高速化について述べる。すなわち本稿は、ICOTにおけるメタ推論システム関連の研究成果の最新報告である。

２．メタ推論

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の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。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。

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、著者ら[7] 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)

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著者らの提案し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[7] 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を制約条件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。例えば、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)

3．メタ推論による制御[15]

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１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。

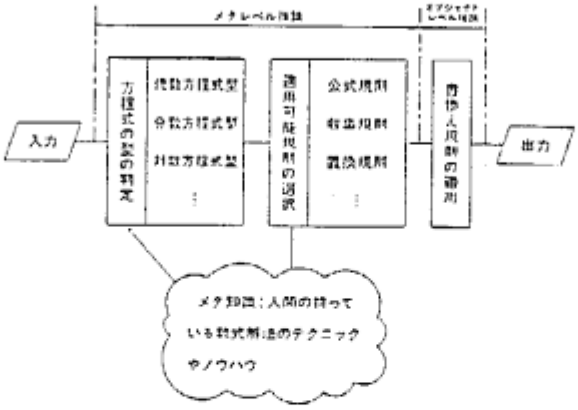

図１　メタプログラミングによる数式処理

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。

```
MD1)  method_demo( Eqn,Ctrl,Ans, _ ,[]):-end_check(Ctrl,Eqn,Ans),!.
MD2)  method_demo( E1 && E2,Ctrl,Ans1 && Ans2,His,H3):-!,
        method_demo(E1,Ctrl,Ans1,His,H1),method_demo(E2,Ctrl,Ans2,His,H2),
        append(H1,H2,H3).
MD3)  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,H3),append(H1,H3,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。MD1)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。MD2)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H1,H2 をappendし、その値をH3に返す。MD3)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与えることができる。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、システムの最適処理実行の構成上重要である。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。知的CAI への拡張についても、今後、検討していく予定である。

４．言語機能拡張機能[12]

人間は複雑な問題に直面すると、もっている膨大な知識および種々の戦略や推論法を使ってその問題を解決しようとする。そのため、人間の知的活動を計算機上に実現する場合には、人間と同様に様々の推論法を自由に実現することが望ましいと考えられる。核言語第１版のたたき台となった並列型論理プログラミング言語Concurrent Prolog (CP)上に構築された知識プログラミング言語Handala [2] は、人間の知的活動を計算機上に実現するために考案された言語で、(1) 論理プログラミング、(2) オブジェクト指向プログラミング、(3) データ指向プログラミング、(4) ルール指向プログラミング、という４つのプログラミング技法の融合を目指して設計された。

さて本章では、新しく拡張された機能の一つであるユーザ定義推論エンジンの組込機能について述べる。ユーザ定義推論エンジンの組込機能とは、ユーザ自身で記述した推論エンジンをHandala の推論エンジンとして組込むことを可能とする機能である。ユーザ定義推論エンジンの組込機能の利点は次の通りであると考えられる。

(1) ユーザ自身で、ユーザ専用の推論エンジンを組込むことが可能となる。
(2) システムとして複数の推論エンジンを容易に提供することが可能となる。

(3) 推論エンジン自体の機能拡張が容易になる。

一般に、オブジェクト指向プログラミングのオブジェクトは内部状態をもち、メッセージによって活性化される。Mandala の実体は，実体への入力列をゴールとみなして解くCPのプロセスと考えることができる。それは次の形で表わされる。

```
instance（<名前>，<入力列>，<世界>）.
```

ここで、<名前>は実体の識別子であり、<入力列>は実体が受け取るメッセージ列である。<世界>は、実体と関係する単位世界名と実体固有の内部状態を保持する。このinstance自身はCPで次のように定義される。

```
instance(Name,[Message | Input],World):-
    simulate(Name,Message,World,NewWorld),
    instance(Name,Input?, NewWorld?).
instance(Name,[],World).
```

第１節のsimulateはMessage をWorld の世界で解き、新しい世界NewWorldを返す述語である。Message の解き方はsimulate述語の定義に従っており、この意味でsimulate述語を推論エンジンと見ることができる。simulate述語の定義を変えることにより、種々の推論を行なう実体を作ることが可能となる。

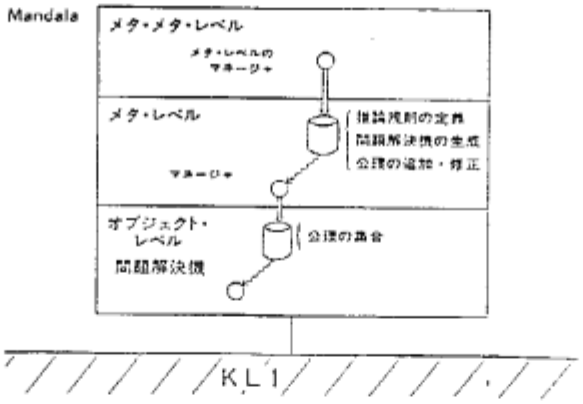

図２　Mandala の構成

図２を用いて、実体の生成方法について説明する。実体を生成するためには、ある単位世界の管理者にcreateメッセージを送る。管理者の単位世界には、create述語の定義以外にinstance述語やsimulate述語などの定義が記述されている。create述語の定義は次の通りである。

```
create(Name,Input,World):-instance(Name,Input?,World).
```

create述語は、instance述語を解くプロセスを生成する。すなわち、生成された実体はこのinstance述語のプロセスに相当し、管理者の単位世界に記述されているsimulate述語に従ってゴールを解く。ユーザが、管理者の単位世界にsimulate述語を記述することができれば、ユーザ定義推論エンジンの組込みが可能となる。以上述べてきたことから分るよ

うに、ユーザ定義推論エンジンの組込機能により、メタレベルのsimulate述語をユーザが組込むことが可能となった。メタレベルでis_a リンクが使えるので、ユーザはsimulate述語だけを記述すれば良い。節を選ぶための述語はシステム提供の述語を使うことができる。すなわち、is_a リンクによりsimulate述語の機能の拡張も容易となる点に注目されたい。

5．知識ベース管理[1,4,8,9]

著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である。そのようなシステムの研究試作の途上で、著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[6,10,11] 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[6,10,11] 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[3] 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[4] 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[6,8,9] 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。著者らは知識ベース管理について、更に次のような課題[5] を検討中である。

(6) 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諸性質を系統的にルール化した。これは高階述語のメタ述語への埋込みによってなされた。
(7) メタ推論を利用して制約用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トリガー述語が有効に適用できることを示した。
(8) 制約用知識の評価実行に種々の可能性を設けるために、評価実行のタイミングを表わす概念を取り入れ、トランザクションという処理単位で統合した。
(9) メタ・レベルの知識同化に伴うオブジェクト・レベルの信念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。

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・パラダイムと知識情報処理パラダイムの融合ということである。

6. 部分計算による推論エンジンの高速化[13,14]

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、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を確立した。本章では、その成果を要約する。

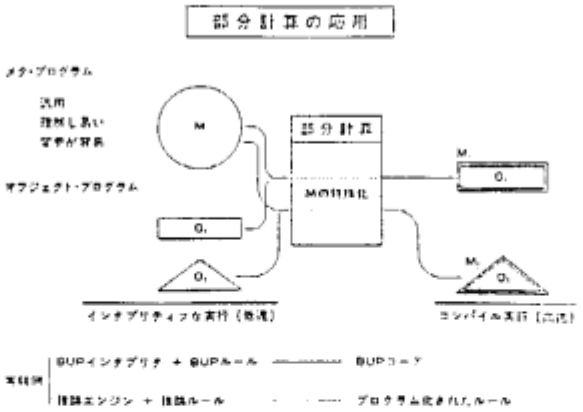

図3　メタプログラムの高速化

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』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。

著者らの提案する Prolog プログラムの部分計算法 PEVAL[13]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3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	901	1157	95
コンパイル実行	110	39	46	25

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

通常、推論システムの作成の方法としては、

(1) 推論部をルール・インタプリタとして作る、
(2) ルールをプログラムへ変換する、

の二つが代表的である。一般に、(1) の方法をとる場合の利点は推論の動きが明確で理解しやすく、かつ、推論部の変更が容易であるということにあり、欠点は推論の速度が遅いということにある。一方、(2) の方法の利点は推論がプログラム化されたルールの直接実行であるから速度が速いということであり、欠点はルールをプログラムへと変換するプログラムが理解しづらいという点にある。この変換プログラムは一般にシンタックス変換、入出力等を含んでいて、推論の戦略がその中でどう表現されているかを見つけるのが難しく、従って、推論に本質的な部分を理解すること、および、その部分を変更することが極めて困難である。

このように方法(1),(2) それぞれの利点、欠点は互いに相補的な関係にある。すなわち、実行効率と推論部の汎用性の間のトレード・オフがそのままそれぞれの方法の利点欠点に現われている。著者らの提案する新しい方法は方式(1),(2) を融合したものであり、実行効率と汎用性の間のトレード・オフを部分計算により解消している。その結果、本稿で提

築する方式はこの両方式の利点を合せ持ち、いずれの方式の欠点も持たない。この方式では、推論システムは方式(1) と同様に推論部と推論ルールに分けて構築される。この段階では推論部は汎用のものが記述され、推論の行なわれ方は理解し易く、また変更も容易である。また特にメタ的な推論を推論部に埋め込むのも容易である。この推論部に推論ルールを与えて実行するときは、推論ルールを固定して、部分計算により推論部を汎用のものから特殊なものへと変換することにより方式(2) の実行効率と同等のものを達成できる。

部分計算を基本ツールとする本方式による推論システムの構築法は、以上のように Prolog 上に作成された既存のものについても有効に適用することができる。また本方式は十分に一般的であるので、推論システムを構築する方法の今後の方向を示唆するものと考えられる。

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、部分計算により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。部分計算のメタプログラミングへの応用としては推論システムの効率化以外にも、任意のプログラムにメタ的制御やメタ的入出力を付加することが考えられる。それは、付加すべき制御や入出力をメタインタプリタで定義しておき、このメタインタプリタをプログラムを固定しておいて部分計算を行なうことによって実現されるのであるが、これについては別の機会に報告する。

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。この一大目標を巧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方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[14]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。

参考文献

1) 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.

2) 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.

3) 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., 131/142 (1984)

4) 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)

5) 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、ICOT TR-　　, 1985.

6) 國藤　進、藤生盛敏、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)

7) 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)

8) 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.

9) 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.

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

11) 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.

12) 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.

13) 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)

14) 竹内彰一、北上　始、古川康一：部分計算によるメタインタプリタの段階的特殊化、Proc. of the Logic Programming Conference '85, ICOT, 1985.

15) 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，1B-4, 29-32, 1984.