解　説

論理型言語指向の推論マシン

1. 論理型言語指向の推論マシンの位置付けと開発の現状†

田　中　英　彦††

1. まえがき

Prolog や KL1 に代表される論理型プログラミング言語は，1971年，A. Colmerauer が試行錯誤過程を含んだ構文解析プログラムを開発したことから発するが，述語論理をベースにした記号処理向き言語で，Lisp に比べて不完全構造が扱えるという利点をもっている．すなわち，扱うデータ構造の中に未知数を含むことができ，値が定まっていなくてもそれを扱うことができる．この特徴によってプログラムの自由度が増し，簡潔で能率が良いプログラムを書くことができる．

一方，推論は，一般に論理の形で与えられた前提条件から結論を導き出す動作を指し，一つ一つそれ自身は簡単な推論を数多く積み重ねて最終の結果を得ることであるが，このような証明過程を与える操作は何も，幾何学における文字どおりの“証明”を与える場合だけに使われるものではなく，大変一般的なものであって，汎用の情報処理それ自身といってもよいものである．したがって，推論を効率良く扱うことのできる機械を作ればそれが汎用計算機にもなり得る．

1982年に始まった日本の第5世代コンピュータプロジェクトは，論理型言語を中心とした情報処理体系を作ることを目的としており，それ以来，多くの大学，研究機関で推論マシン研究が行われてきたが，その多くが論理型言語を対象としたマシンである．したがって，ここではこのような論理型プログラミング言語を効率良く処理する機械にしぼり，その概要と位置付けについて述べる．

2. 推論マシンの位置付け

2.1 推論と述語論理

（1） 推　論

なんらかの問題が与えられたとき，それを解こうとする試みは，われわれの日常生活に深く根ざした作業であるが，論理学や人工知能の分野ではこれを問題解決と呼んでいる．問題解決の過程は広い意味での推論 (inference) の過程と考えられ，したがって推論は非常に基本的な操作であるといえよう．

推論の最も基本的な形式はいわゆる三段論法であるが，推論にも幾つかあって，前提事実と規則から結論を導き出すものを演繹と呼び，結論と規則からその前提事実を導く操作を仮説形成という．これらは，別名前向き推論，後向き推論とそれぞれ称するが，これらのほかに，前提事実と結論からなるペアを多く集めたものからこれらに共通する規則を導き出す操作も存在し，それを帰納推論と呼んでいる．

前の二つが，機械的な操作として実現することが比較的容易であるのに対し，帰納を実現することは一般に非常に難しい．したがって，推論処理というとき，それはほとんどが演繹や仮説形成であり，ここでもその意味で推論という言葉を用いる．

推論を進めてある結論を導くことは，幾つかの論理式からある命題が導けることを証明することと等価である．この操作は通常，証明すべき命題の否定を元の論理式に加え，その矛盾を導き出すことによって行う．

一般に，論理式の無矛盾チェックは，出現する変数にあらゆる値を入れて試すことによって行えるが，それではあまりにも能率が悪い．これに対する解答が，Robinson の導出原理であり，これによって証明は現実的な操作となったのである．

（2） 述語論理とプログラミング言語

human (socrates) のように書いて，「socrates は human である」と読み，socrates のところに，変数 X を入れて，human(X) を，X の値によってその真偽の定まる関数のようにみなしたとき，

† Logic Programming Oriented Inference Machine by Hidehiko TANAKA (University of Tokyo, Department of Electrical Engineering).

†† 東京大学工学部

human を述語記号と呼ぶ．このような表現を基本的な論理式として組み合わせたものが，述語論理である．たとえば，

human(X)−＞mortal(X).

は，左辺を満たせば右辺が成立すること，すなわち，X が human であれば，それは mortal である（いつかは，死ぬ）ことを述べている．

このような述語論理式のうち，組合せ方を制限し，証明手続きとして効率的なものが存在するものに限ったのがプログラミング言語としての述語論理である．具体的には，矢印の右側に現れる述語を一つ以下に制限したホーン節がよく用いられる．

たとえば，階乗計算を例に取ると，

$f(0, 1)$.

$N1=N-1$ and $f(N1, X1)$and

$X=N*X1->f(N, X)$.

の二つが，階乗を与える論理式で，$f(0, 1)$ は，$0!=1$ であることを表し，二つめは，$N!$ の値を X として解くための条件を，$(N-1)!$ との関係として与えたものである．今，$f(3, X)$ を求めたい $(3!)$ とき，これと上の一般式から

$f(2, X1)$and $X=3*X1->f(3, X)$

$f(1, X2)$and $X1=2*X2->f(2, X1)$

$f(0, X3)$and $X2=1*X3->f(1, X3)$

が次々と得られ，最後の左辺に現れる $f(0, X3)$ は，一つめの論理式から $X3=1$ として定められる．すなわち，証明が与えられたわけで，それと同時に，X の値も，

$X=1*2*3=6$

と与えられる．

このように，証明手続きとプログラム処理過程とは密接な関係があり，論理式の集合は，一般にプログラムとみなすことができる．

2.2 推論マシンの構成

ホーン節に基づく論理型言語を効率良く処理する機械が推論マシンであるが，これには二つのものがある．

一つは，論理型言語の処理系を，Cなど，通常のプログラミング言語で記述して実現し，それを通常の計算機の上で動かす形の推論マシンで，厳密に言えば推論マシンとは言い難いが，広く考えればこれを推論マシンと呼ぶこともできよう．ほかは，論理型言語の処理に向いた形の専用計算機としての推論マシンである．前者が処理の速度をソフトウェアの実現方式の工夫と，マイクロプロセッサなどの素子技術の発達によって向上することを期待しているのに対し，後者は，それらに加えるに，推論処理のハードウェア支援を考えているところに特徴がある．

また，処理系を実現する方法にも二通りあり，処理系をインタプリタで実現したものと，なんらかの機械語に落として処理するコンパイラ形式のものがあり，インタプリタ方式は実現が容易で対話的扱いに優れているのに対し，コンパイラ方式はインタプリタ方式よりも数倍早いのが普通である．

さらに，マシンの実現として，単一のプロセッサを用いた逐次型推論マシンと，複数のプロセッサを用いた並列推論マシンとがあり，処理対象とする論理型言語の種類がこれらに合わさって多くの推論マシンが存在する．

2.3 推論マシンの位置付け

述語論理自体は汎用の処理であって，種々の応用に用いることができる．しかし，プログラミング言語としては，非決定性を含んだり，動的なデータ構造を扱うことができるようになっているため，一般的には記号処理に向いた言語と考えられる．

科学技術計算を高速に処理することを目的としたスーパコンピュータは，パイプライン方式に基づいているため，処理が決定的であることが必須であって非決定的な処理にはあまり向いていない．しかし，自然言語処理や，エキスパートシステムなどでは，そのサイズや構造が処理の最中に変化する動的なデータを柔軟に扱ったり，複雑な制御の流れを適切に扱うことが必要で，論理型言語はそのような応用には大変向いた言語である．

また，データベースとの関連をみれば，関係論理は，論理型言語と大変相性が良く，プログラミング言語とデータベースを融合したシステムを実現する優れたベース言語となりえるし，将来，記号処理の分野が大きく発展するであろうことを考えれば，その時点で現在の数値処理やデータ処理を統合することが考えられ，その場合に中核となるベース言語体系として期待できよう．

すなわち，問題が与えられたとき，その都度適当なプログラミング言語でプログラムを作る現在

のソフトウェア体系から，さまざまな知識を少しずつ蓄えていき，問題が与えられれば，それらの知識を組み合わせて問題を解決する形の知識情報処理への移行である．この処理体系では，知識の組合せのほとんどを計算機自身が行い，プログラムを知識として与えることが，プログラマの役目であって，将来は知識の獲得を自動化することも考えられる．

そのような世界からみれば，推論マシンは，現在の記号処理専用から，あらゆる処理の基盤としての汎用マシンとなるであろう．これこそ，日本の第5世代計算機プロジェクトが目指している目標である．

3. 推論マシン研究の流れ

3.1 論理型言語の研究

論理型言語の最初は Prolog であるが，その後，R. Kowalski らによる，Prolog のプログラミング言語としての解釈，D. H. D. Warren による効率良いコンパイラの開発，第5世代プロジェクトによる本格的取り上げを経て，大きくクローズアップされるところとなった．

論理型言語では，もともと，多くの論理式間の単一化を試行錯誤的に探索する操作を含み，幾つかの候補を次々と探す処理が重要である．これは並列にチェック可能であり，したがって，並列処理に向いている．

Prolog でもその探索処理があるが，Prolog ではそれを後戻りによって実現しており，逐次的に処理される．しかし，より本格的な並列処理を目指して，並列論理型言語の研究が1980年ごろより始められた．この研究の流れは二つに分けられ，Prolog を中心としたものと，Committed Choice 型言語（CCL）と呼ばれるものがある．

（1） 並列 Prolog

Prolog のシンタクスはほとんど変えず，その処理方式を並列化したものがこれである．もともと，Prolog を述語論理として解釈すれば，論理式間の処理順序はまったく自由なはずで，それを Pure Prolog と呼ぶ．並列探索としては，AND-OR Process モデル，独立 AND 並列モデル，Restricted AND 並列モデルなどがあり，AND-OR Process モデルは，Pure Prolog に現れる OR 並列と AND 並列とを一般的に扱ったものである．

一般に，OR 並列は実現が容易であるが，AND 並列は共有変数の存在によって，処理が複雑となり，オーバヘッドが大きくなりがちであるため，完全に実現することはせず，そのような問題が発生しない場合のみに限定して並列処理をするのが普通であり，独立 AND や Restricted AND はそれである．

言語としては，通常の Prolog に少しの追加をして，並列部分の明示的指示をする形のものと，できるだけシンタクスを触らず逐次型のものとの互換性を重視したものがある．

（2） Committed Choice 型言語

論理式間の一致を取って解を探してゆく過程において，幾つかの候補が見い出されたとき，Pure Prolog ではそれらの候補すべてについて，並列に探索を進めるが，候補のうち，どれか一つのみを選択し，ほかを捨てて処理を進める方式が，この Committed Choice 型言語である．したがって，構文中には，その単一選択を行う場所を示す Commit bar があるのが特徴である．

この種の言語の源は K. Clark らによる1981年の Relational Language で，その後，Concurrent Prolog, Parlog, Oc などが開発され，1985年には ICOT の上田によって，Guarded Horn Clauses が開発され，処理系の効率化が達成されるに及んで，第5世代プロジェクトでは，GHC にほかの必要機能を付加した言語 KL 1 を中核言語として採用し，それ用の処理系，OS，ハードウェアなどを開発している．

3.2 論理型言語処理系の研究

Colmerauer が作った Prolog は，最初あまり速くなく，また所要メモリも比較的大きなものであったが，D. H. D. Warren が，1977年に作った処理系は，処理速度が十数倍，所要記憶域が10分の1と著しい改善をみせ，Prolog 実用化へと大きく踏み出す原動力となった．

これは，単一化のコンパイルと，変数の種類によって用いるスタックを分けたことによるものであり，DEC-10 のアーキテクチャ上の特徴をうまく利用した処理系で，DEC-10 Prolog と呼ばれている．

この後，第5世代プロジェクトで Prolog が取り上げられ，KL 0 言語をハードウェアの支援により高速に処理する推論マシン PSI が作られた．

また，幾つかの処理系も作られた．

1982年，Warren は，VAX のようなレジスタベースの汎用マシンを対象として，DEC-10 Prolog よりもより優れたコンパイル技法を発表した．これは，Warren Abstract Machine Instraction Set (WAM) と呼ばれる仮想機械命令セットを設定し，Prolog をそれにコンパイルするもので，その仮想機械命令のインタプリタを作りさえすれば，どんな計算機の上にでも走らせることができる．

この方式のポイントは，通常のプログラムでは後戻りが少ないことを利用して，できるだけその情報をスタックに積まないようにし，引数を置いておくレジスタを設け，その割り当てを最適化してメモリへのアクセスを減らすことにある．

この方式は大変成功し，その後の推論マシンはほとんどこの種の方式を採用するところとなった．もともと，WAM は，逐次型計算機の上で走らせることを想定して作られたものであるが，並列推論マシンにおいても，基本は変わらず，幾つかの並列処理支援用の命令を追加して使われているのが現状であり，単に Prolog に留まらず，Committed Choice 型言語に対しても同様である．

3.3 推論マシンの研究

（1） 逐次型推論マシン

Prolog の特徴的な処理は，単一化と後戻りである．これらを実現するためには，データ型のチェックとスタックが有効であるが，後戻りがあるため，スタックには単なる LIFO の機能だけでは不十分で，スタックの浅い所のみならず深い所をもアクセスする必要がある．したがって，キャッシュのようなランダムアクセスをもったメモリのほうが，ハードウェアスタックよりも有効な場合がある．

逐次型マシンは，当初，まず，Prolog を直接実行する高級言語マシンを実現する形での実現が検討された．すなわち，ICOT の PSI であるが，まず，KL0 をいったん表形式の中間言語に落とし，それをマイクロプログラムで解釈するものであった．結果は，解釈のオーバヘッドが大きく必ずしも成功したとは言えないが，Prolog の振舞を知る上で大変役立ったと言えよう．

その後の逐次型推論マシンの研究は，WAM の出現によって大きく影響を受け，ほとんどが WAM ベースのコンパイラ型システムになった．したがって，ハードウェアの支援としては，WAM の効率的実行を支援するもので，タグアーキテクチャ，レジスタファイル，3段程度のパイプライン処理，キャッシュ，複数のメモリアドレスレジスタ，などである．

（2） 並列推論マシン

並列マシンの研究は，まず Prolog の並列化から始まった．AND-OR Process モデルが最初であるが，AND 並列の実現をオーバヘッド少なく行うのが難しいことによって，まず OR 並列のみを対象とした研究が行われた．並列マシンの研究は，まず計算モデルの検討から始まり，OR 分岐したときにオーバヘッド少なく情報をほかのプロセッサに引き渡すためのデータ共有／コピー方式の検討，多くのプロセッサがあった場合の負荷分散の方法などに努力が注がれた．

しかし，OR 並列だけではプログラミング言語としての記述性に問題があるため，AND 並列を取り込む努力が始められた．ここで世界の研究は二つに分かれる．一つは，Committed Choice 言語の進展に従って，それを対象言語とした並列マシンの研究であり，ほかは，従来型 Prolog の実行に AND 並列を入れ込もうとする努力である．

前者の研究は，ICOT を中心として行われ，最初は Concurrent Prolog から検討が開始され，続いて GHC をベースとした KL 1 に発展していった．これらの研究は，現在も進行中で，その成果は，第5世代プロジェクトのプロトタイプとして，1991年度末には提示される予定になっている．

後者の研究は，日本においても当初行われていたが，現在は海外にその中心が移った．実行が容易な AND 並列のみを明示的にまたはコンパイラで自動的に取りだし，実行するものである．研究の中心は特殊ハードウェアの研究よりも処理系の研究にあり，Symmetry などの汎用並列計算機（マルチマイクロプロセッサ）を用いてその上で研究をする形がほとんどである．一部，計算機アーキテクチャの検討も行われているが，逐次型マシンのプロセッサの検討とそれを要素に用いた並列マシンの研究，もしくは紙上検討のものである．

4. 推論マシンの開発状況

4.1 逐次型推論マシン

推論マシンの研究は，まず通常の汎用計算機の上に処理系を載せることから始まった．したがって，これらは，推論マシンというよりも，コンパイラの研究といったほうが適切であろう．

推論処理に対するハードウェア支援をも考慮した文字どおりの推論マシンの研究は，前述のとおりICOTのPSIから始まり，タグアーキテクチャ，レジスタファイル等に関する知見が得られている．

PSI以降の研究は，ほとんどがWAMベースの研究である．ICOTのCHI, PSI-II, CHI-II, カリフォルニア大学のPLM, 三菱電機のPegasus 東芝のAIP, 日立のIPP, 欧州コンピュータ研究所（ECRC）のICMなどで，WAMを改良し，新しいコードを付加したり，除去すること，さらにハードウェア構造を工夫することによって，Append処理の性能でいって，200 KLIPS (Kiro Logical Inference Per Second) から1 MLIPS程度の性能を得ている．このうち，IPPがミニコンにタグ機構などを付加して強化し，通常の言語処理との共存を図ったものであるほかは，すべて専用のチップないしは，専用ハードウェアボードを用いたシステムである．

その後，推論マシンの研究は，コンパイラの研究に戻ってきた．一つは，WAMコードを汎用機の上で実現した処理系が数MLIPSの性能を出し始めたことと，RISCに代表されるマイクロプロセッサの著しい発達により，特殊なハードウェアを開発するよりも，マイクロプロセッサ上にソフトウェアで処理系を実現するほうが，その時点での絶対性能が高いものを容易に得ることができるからである．

RISCは，コードサイズが大きく，メモリアクセスの頻度がCISCよりも高いため，通常処理よりもメモリアクセスに対する要求の高い推論処理に必ずしも適しているとは言えず，処理応用によってその性能差が出がちであり，また，実行時の型判断を必要とするような処理に対してはパイプラインの乱れが大きくなり，RISCの高速性能を生かし難いが，新しいプロセス技術をいち早く取り入れることのメリットや，進歩するコンパイラ技術によってその性能としてはかなり高いものが得られるのが現状である．

4.2 並列 Prolog マシン

並列Prologマシンの研究は，AND/ORプロセスモデルの後，ICOTを中心として処理系やシミュレータの研究が行われた．PIM-P, PIM-D, PIM-R, 神戸大のK-Prolog, 東大のPIE, SICSのOR-parallel token machineなどである．

これらはいずれもOR並列処理モデルを中心とした研究で，ソフトウェアシミュレータやハードウェア試作をとおして，データや，コードの共有／コピー方式，効率良い負荷分散方式などの成果が得られている．

AND並列の研究としては，神戸大のPARK Prologなど，並列処理を明示的に記述する方向の研究も行われた．また，ハードウェアとしては，データフローマシンを用い，その上で並列Prologのコードを乗せて動かすという検討も行われた．PIM-Dや通研のDFM-2がそうである．

その後，ICOTを中心とした研究はCommitted Choice言語を対象としたものに移ったため，単なるOR並列の研究は日本の中では少なくなったが，いまだ上記ポイントに関して検討すべき余地もあり，何カ所かで研究が続けられている．

また，英国，米国，スウェーデン，ドイツを中心とした研究では，OR並列からスタートし，Prologのシンタクスをできるだけそのままに，並列性をできるだけ取り出そうという研究が続けられている．

すなわち，独立AND, Restricted ANDなど，やさしいAND並列だけを取り出す方式や，さらに進めて通常のAND並列をも利用しようとするAndorraなど，AND/OR並列モデルの研究が盛んである．また最近は，これに制約プログラミングの機能を付加し，より記述性に優れ，処理効率の良いシステムにしようという研究がある．

これらの研究のポイントは，効率良い処理系の検討が中心であるが，Warrenのグループでは，多くのプロセッサを結合し，データがキャッシュのように自動的に必要なプロセッサへ拡散するようなData Diffusion Machineという構想も検討されている．

4.3 CCL 向き推論マシン

OR並列マシンは，なるほどある程度の並列性を出すことができるが，それだけでは並列性を

取り出すことのできる問題が限られる．したがって，期待はAND並列をいかにして取り出すかによせられた．その結果，Committed Choice型言語中でも実現オーバヘッドの少ないGHCを中心とした言語が対象となった．

研究の中心はしたがって，ICOTであり，GHCをベースにした核言語KL 1の検討から，その実装，それ向きの並列マシン，オペレーティングシステムなどの研究が行われている．そのほか，言語としてはImperial CollegeのStrand，GHCにOR並列性を導入したICOTのANDOR-IIなどの研究もこれに属する．

研究のポイントは，効率良い処理系の開発ということのほかに，並列マシンであることからくる幾つかの項目がある．すなわち，CCLは一般にメモリに対するアクセス頻度が高く，また，比較的処理の粒度も小さいので，並列プロセス間の通信要求が高く，それが処理のネックになりやすい．したがって，いかにして，通信を減らしオーバヘッドを低減するかということと，効率良い負荷分散方式，どのプロセスから処理を始めるべきかを定めるスケジューリング方式（これは，CCLのように，実行を中断させる機構の備わった言語では，処理効率の善し悪しに大きな影響をもつ．）などにある．

作られたマシンとしては，PSIを数多く接続したMulti-PSI第一版（6台），第二版（64台），東大のPIEEEE，などがあり，現在試作中のものにICOTを中心とした5種類のPIM，東大のPIE 64などがある．Multi-PSIは，最初の大規模な実験機として作られ，これをベースに言語処理系やオペレーティングシステムの開発が行われたほか，多くの応用を乗せて実験が行われ，さまざまな知見が得られている．

たとえば，従来，複雑な並列プログラムは大変作成が困難であると言われてきたが，大規模なOSであるPIMOSの作成経験や，応用プログラム作成経験から，従来の困難さの原因であったプロセス間の同期は，並列論理型言語の場合，ほとんど問題にならないと言われている．

5. むすび

Prologや KL 1など，論理型言語をその処理対象とした計算機，推論マシンの研究について，その位置付け，歴史，研究の流れ，技術のポイント，研究動向などについて述べた．推論マシンは一見特殊目的の専用マシンにみえるが，その中身は大変一般的な処理機械であって，新しい計算機の基本形であると捉えることもできる．数値処理を基本とした従来型計算機はそういう観点からみれば，特殊マシンであり，大きな新しい枠組からみればその一部分を占めているに過ぎないともいえよう．

一方，並列処理という観点からみたとき，並列処理では，そのプログラミング手法がいまだ未成熟であり，さまざまなプログラミング支援環境の整備が必要で，処理の稼働状況を視覚的に見せるツール，プログラムの並列性を解析しチューニングするツール，プログラムのバグを探すツールなどを備えることが重要である．

これらのマシンが今後どう発展してゆくかは，論理型言語の広い普及と，並列処理の優れた支援環境の整備にかかっている．現在，ICOTでは，第5世代コンピュータのプロトタイプとして，1000台規模の並列推論マシンが作られており，ここ1〜2年の間に稼動を始めることが期待されている．論理型言語という新しい枠組の上に，最新のハードウェア技術を用いて作られているこの機械は，長い目でみて情報処理の世界に大きな影響をもたらすことであろう．

参考文献

1) 田中：非ノイマン型コンピュータ，電子情報通信学会編（1989年11月）．
2) 特集「第五世代コンピュータ」，人工知能学会誌，Vol. 4, No. 3（1989年5月）．
3) 特集「AIマシン」，人工知能学会誌，Vol. 2, No. 4（1987年12月）．

（平成2年10月5日受付）

田中　英彦（正会員）

昭和18年生．昭和40年東京大学工学部電子工学科卒業．昭和45年同大学院博士課程修了．工学博士．同年東京大学工学部講師．昭和46年助教授，昭和62年教授．昭和53〜54年ニューヨーク市立大学客員教授，現在に至る．計算機アーキテクチャ，並列推論マシン，知識ベース，オブジェクト指向プログラミング，分散処理，CAD，自然言語処理等の研究を行っている．「計算機アーキテクチャ」，「VLSIコンピュータI，II」，「ソフトウェア指向アーキテクチャ」（いずれも共著），「情報通信システム」著．電子情報通信学会，人工知能学会，日本ソフトウェア科学会，IEEE，ACM各会員．