平成元年12月20日発行
毎月1回20日発行　通巻第9号
平成元年5月16日第三種郵便物認可

1

1990 JAN

コンピュータアクセス

お正月特集
パソコンの資産
- ハード、ソフト、周辺機器の紹介
- 快適なパソコン利用のために
- こんな場合どうする

解説
リレーショナル・データベース

CONTENTS

コンピュータ アクセス
1月号

1990. JAN

投稿

ソフトウェアをコンピュータに作らせる夢
—1つの提案「MIS」について—

㈱インターナショナルソフトウェア
研究開発部　中村三郎、田代達也
文教大学　情報学部　鈴木昇一

ソフトウェアのバックログの解消はいつのことか

そもそも〝コンピュータ〟とは仕事をして人間の仕事の肩がわりをする機械なのである。コンピュータにはソフトウェアが必要なのは周知のとおりだが，ソフトウェアの作製は規模が大きくなると大変な手間と時間がかかる。しかも，我々人間プログラマは一朝一夕には技術移転が難しく，プログラマが足りないというのが現状である。

そこで，何とかこのソフトウェアをコンピュータに作らせられないものか，というテーマのもとに，近年様々なアプローチが行なわれている。

プログラムを作る方法としては，次の3つが考えられる（作るのが人にしても機械にしても）。

1．仕様からプログラムを生成
人の手によって最も一般的に行なわれているもので，プログラム言語知識と論理化技術及びハードウェア知識を駆使する。どれか一つ欠けていてもプログラムは動かない。

2．事例からプログラムを生成
人はこれを連想とか類推とかの試行錯誤で行なうため，経験的知識がものをいう。

3．仕様そのものがプログラム
形式的仕様記述が厳格で，作るのが難しい（自然言語ではまだ不可能）。

人がプログラムを作る場合はもちろん，機械が作る場合（＝ソフトウェア自動生成システム）においても仕様を入力しなければならない。機械に作らせる場合の仕様記述は，文章的であれ図形的であれ，一種の超高級プログラミング言語によって行なわれる。現在，ソフトウェア自動生成システムに入力すべき仕様は，その記述が複雑で難しく，微細にわたるため，極めて限られた人しか記述できない。しかも，従来のプログラミング言語で書いたほうが容易な場合もあり，敬遠されがちである。

理論サイド（特に知識工学的アプローチ）では，大まかには先の1～3の角度で，かなりの成果をあげている。しかしながら，そのいずれもソフトウェアの生産現場から見ると実体はまだおもちゃのレベルか，「論」の域を出ていないのが現状である。

一方，ソフトウェア工学（ソフトウェアに対する工学的技法）は，まずソフトウェア生産の工程別方法論や技法を考案し，次に一貫生産性実現のためにソフトウェア生産工程間の連係を図る，といったアプローチであり，プログラムを人が作る上での生産的な方法論の研究だといえる。

1950年代や1960年代には，コンピュータ・プログラミングはひとつの技巧とみなされ，工学的アプローチは行なわれなかった。ところが，ソフトウェアはコンピュータ利用システムの数，複雑さ，適用域に比例してその需要が増大し，社会的重要度が高くなってきた。そこで，ソフトウェアをより効率良く生産することを目的としてソフトウェア工学が生まれ，現在に至っている。

Σシステムも名前の由来にあるように，ソフトウェア生産工業化を実現させ，ソフトウェア開発作業を職人芸的な産業から装置産業へ脱皮すべく頑張ってはいるが，言い方を変えれば，コンピュータにソフトウェアを作らせることが，現時点では出来ないことによる次善策ともいえる。

このような話をすると，そんなに手間がかかるのならプログラムをできるだけ作らないのが一番いいのではないかと言う方もいると思うが，そういうアプローチから再利用とか部品化ということの有効性が認識されるようにな

ってきた。つまり，新規に作らねばならない部分をできるだけ小さく抑えるという方法である。

しかし，それはそれでまた新たに部品の整備や再利用可能化のスケール，パワーといった問題が生じる。ただそういったアプローチによって，新規のプログラムを作らなければならない割合は以前に比べて減ってきていることは事実である。

このように，ソフトウェア生産性の向上にはいくつもの障壁がある。しかし，ソフトウェアをコンピュータに作らせるという「夢」は追いたいものである。先に述べたようにおもちゃのレベルではあるが，これから本題で紹介する「MIS」なるものは，実はこの夢の実現を原理的に保証しており，その意味では数ある理論的成果の中でも，ひときわ魅力あるものと言える。

通常はプログラムを作ってから，そのプログラムが正しいかどうかをテストするが，次で紹介するモデル推論システム「MIS」では逆に，意図するプログラムのテスト例（プログラムの入力と出力の対の事例）を多数与えて，プログラムを自動合成させるのに使用できる。つまり，MISで作られたプログラムは事例については検証済みのため，テストレスである。

MISの紹介

コンピュータにソフトウェアを作らせる試みは，研究レベルでは多方面からなされており，例えば次のようなものがある。

①従来からのコンパイル方式の延長上にあるもの

②ソフトウェアの型紙のようなものから可変枠を埋めこむもの

③ソフトウェアの部品を合成するもの

④ソフトウェアの部品を組合せるもの

プログラム自動合成の範ちゅうで考えれば，MISはこのうち③の部分に属するものである。MISは，帰納学習によって事例（入出力例）からソフトウェア部品を合成する能力がある（というより現時点での能力を考え併せれば，もう少し謙虚に，原理的な枠組の提示と言った方が良いかもしれない。また，与える言語やでてくるプログラムの言語はPrologを対象にしている）。

帰納学習システムには種々の定式化があり，その一つに“ある対象領域について観察した結果から，一般的な原則を発見して，その規則をもつ世界のモデルをつくり出すようなプロセス”がある。帰納学習システムの実現は人工知能研究の興味深いテーマであるばかりでなく，エキスパート・システムの実現において最も難しい問題となっている（Feigenbaumのボトルネックと呼ばれるところの），知識獲得の自動化にも大いに役立つと考えられている。また，（入出力例からの）プログラム自動合成が部分的にでも可能となれば，それは現在もっぱら人手に頼っているソフトウェア作成のコストを下げる画期的な手段となるのでは，と期待されている（そうは言うものの，後でも述べるが，複雑なものの事例の与え方が問題となり，かつ実際に意味のある程度の学習には，一般に仮説空間内の探索木を少しずつ生成しながら大量の仮説探索を必要とする）。

MISは帰納的モデル推論システム（Model Inference System）の略であり，真あるいは偽と判明している事例が次々と与えられると，与えられた事例はもちろんのこと，与えられなかった事例の真偽をも決めることができるメカニズム，仕掛け，モデル，を帰納的に推論するシステムである。具体的には，与えられる事例に対する人間からの真偽の回答から，Prologプログラムを生成する（MISは1981年E.Y. Shapiro博士による論文「Inductive Inference of Theories From Facts＝事実による理論の帰納的推論」で提案されている）。

プログラム生成の過程は，仮説の設定と検証を繰り返し，すべての事例を証明できる仮説を作り上げていくものである。AI流に分類すれば「知識を自動的に獲得するシステム」であり，「事例から一般的な規則を学習するシステム」になる。

ここで次章「MISの仕掛け」にいく前に，帰納推論は初めてという方のためにプログラム作成上の要素に対比させる方法で説明したい。「事例」はテストデータ，「仮説」はプログラムになる候補，「検証」はテストのことであり，「オラクル（＝神託）」とはプログラム機能仕様を知っている人や，設計者，仕様要件を要求するユーザーのことを示す（その人からの回答事例を指す場合もある）。

MISには，事例や仮説を表現する言語があらかじめ提示されて，MISはその言語の範囲で仮説の生成を行なう。

オラクルはMISに事例の無限列を供給する（無限といっても，一体いつまでかというより，必要ならいくらでも事例を供給する能力があると考えてほしい）。

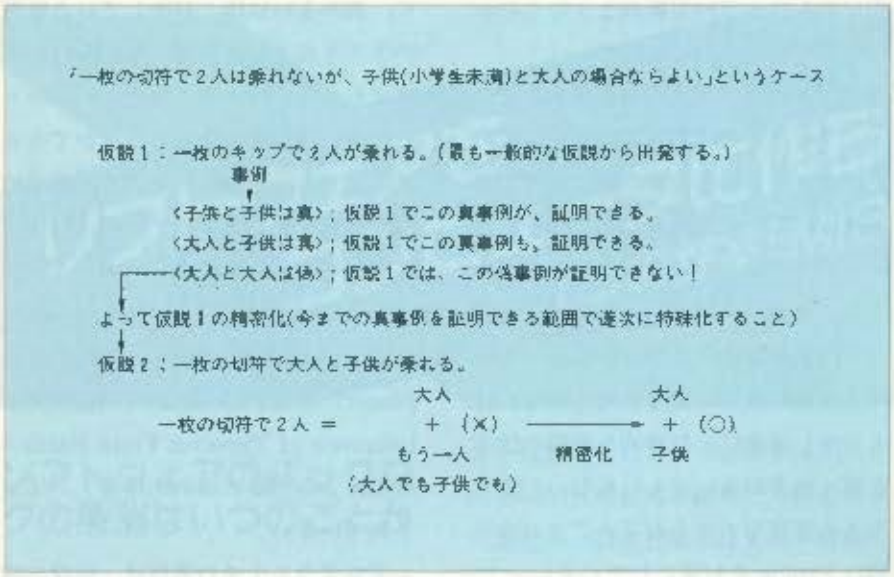

図1.1 MISでの精密化による仮説生成例

事例は
〈文(ある事実の言明)，真／偽の値〉
の組の形で与えられる。
〈〉の中の左を観測文，右をその真理値と呼ぶ。例えば，
〈(2＞5)，偽〉
とか
〈彼女は美人である，真〉
といった具合である。

理論では真の事例を正事例，偽の事例を負事例と呼ぶ場合が多いが，ここでは対応づけしやすいように，それぞれ真事例，偽事例と呼ぶことにする。MISの目的は操作的に述べれば，次々と到着する事例の列を読む都度，偽事例を偽と反駁できない誤った仮説の除去と，以前に除かれた誤った仮説の修正で派生するものを付加することを繰り返し，その過程で今までの全ての事例を矛盾なく説明できる真なる仮説の集まり（＝プログラム）に到達することである。空の仮説から出発して，与えられた言語で記述し得る仮説空間内の仮説探索を，その全体を調べ上げずに行なうため，効率良く進めてもいる。誤った仮説の発見部分を「矛盾点追跡」，仮説の生成部分を「精密化」と言う。図1.1にMISでの精密化による仮説生成例を示す。

MISによる生成プログラムは，図1.1のように偽事例を真と導いてしまうプログラムよりも特殊化したプログラムになる。帰納学習においては真事例を真と導けないプログラムよりも一般的なプログラムを得る為に，精密化とは逆の処理（汎化）を使ったものもある（後述「MISのこれから…正事例の活用」の項に紹介）。

では，MISは帰納学習においては処理が不充分かと言うと，そうではなく，ある仮説の最も一般的なところから，その仮説の特殊化の方向での全体集合に向かって仮説候補の枝刈りの効率化を企てる意味の矛盾点追跡と，精密化のアルゴリズム（特殊化した仮説を付加して得られる仮説集合のうち一般的で簡単なものを得る方法）を提示している。

MISの仕掛け

本章は今までの「夢」の話の理論的なところであり，Prologを知っている人に向いている。MISの論理の説明の前に，用語の定義と目的を説明する。

定義

1. ある「言語L（一階述語）」が与えられている。
2. 観測言語Lo(事例の組の左部分の観測文を記述するのに用いる言語）と仮説言語Lh(仮説を表現するための言語）は言語Lで表わされ，自動生成するプログラムも言語Lで表せるものとする。
3. 観測文（Loの文）aの真偽を回答可能な人間をオラクルと言い，MISにとっては，人間に質問し回答を読む操作も，事例を読みこむのと同じく知識を得ることである。
4. 事例は観測文aと真偽Vの対で表す（第i番目に与える事例Fi＝〈a，V〉でV＝true／false)。

目的

1. 事例を読みこみ，仮説言語Lhの文の有限集合を出力すること。MISはモデルについて有限個の事例を確かめ，間違った推測を有限回行なった後，事例を矛盾なく説明できる真なる仮説の集合を正しく推論できることが理論的に保証されている。得られる文の集合が，求めるべき（Prologの）プログラムである。

記法説明

1. A，B，Cなどを論理記号を含まない論理式（原子論理式）の名前とす

る。

2. 観測言語Loの各観測文はAに含まれる変数がX，Yとすると，A（X，Y）の各変数に具体的な値を代入した形のもの（グランド原子論理式と呼ばれる）となる。

3. 仮説言語Lhの各仮説はA：－.（事実節），A：－B，C.（規則節）の形をしている確定節である。記法はDEC-10 Prologの記法を採用し，各所で「文」というのはPrologの節。

4. Strueは真事例の集合，Sfalseは偽事例の集合とする。

5. 仮説の節集合をHとし，H中の節で“false”のマークが付けられていないものの集合をHtrueとする。

MISの論理

MISのソフトウェア内部構造は，機能的に次の2つに分けることができる。1つはProlog処理系で，もう1つはMISエンジンである。Prolog処理系の役割は真事例や偽事例の真偽の検証で，いわば，仮説の検証装置のようなものと言える。MISエンジンは次の3つの部分が機能分担して動く機能推論の本体である。

1. モデル推論システム制御部（MISメイン）
仮説生成と検証を制御し，事例の到着毎に逐次帰納推論のステップを進める部分。

2. 矛盾点追跡
偽事例を真と導出した過程を追跡調査して，誤った仮説を発見する部分（プログラマが行なっている虫取り作業）。

3. 精密化
真事例を説明する仮説を立てる部分。仮説生成はいろいろ改良を要するところで，最も基本的なやり方では，Prologの文法をベースに確定節の複雑さとかを尺度に，一歩一歩進める方法がある（この部分でプログラムが作られる）。

次に，この3つの部分における各処理について説明する。

モデル推論システム制御部

ここではまず，

- 真事例の集合　Strue＝｛　｝
- 偽事例の集合　Sfalse＝｛□｝
- 仮説の節集合　H＝｛□｝

と初期設定を行ない（□は空節を表す。H中の□には“false”のマークを付加しておく。），次の(1)～(4)の処理を行なう。

(1)事例の逐次的とりこみ，事例の(真／偽)による振り分け。新たな事例(観測文とその真偽の組）を一つ読み，その真／偽に応じて，Strueあるいは Sfalseに加える。

(2)次の二つのWhile文の( )内の条件が共に成立しなくなるまで，以下の操作を繰返す。

〈－1st while－〉

While（Sfalseに属す事例がHtrueから導出できる）

［1st whileの脱出条件の検証］

偽のマーク付けのない仮説の集まりHtrueから偽事例（Sfalseに属す事例）が一つでも真と導けないこと。

［1st whileの体部処理］

それまでに読みこんだ全ての偽事例が真と導けなくなるまでの次の処理を繰返し実行。矛盾点追跡モジュールによって，その原因であるHtrue中の仮説を求め，それに“false”とマークしHtrueから削除する。つまり，ある偽事例が真と導けた場合，そのときの導出過程を追跡調査して，誤った仮説に偽のマーク付けをする。

〈－2nd while－〉

While（Strueに属す節がHtrueから導出できない）

［2nd whileの脱出条件の検証］

偽のマーク付けのない仮説の集まりから真事例が全て真となること。

［2nd whileの全体処理］

それまでに読みこんだ全ての真事例が真と導けるまで，次の処理を繰返し実行。Strueの要素でHtrueから導出できない真事例をQとする。H中の“false”のマークが付いている節を精密化アルゴリズムによって精密化し，それとHtrueとを合わせたものから真事例Qが導出できるものを探し，その精密化された節をHtrueに加える。つまり，ある真実例が真と導けなかった場合，以前に偽のマーク付けされた仮説を精密化する。

(3)1st while & 2nd whileともWhile文の条件が共に成立しないで，無事何事もなく脱出したかどうか確認する（新たな偽のマーク付けもなく精密化もない状態でwhileループを脱出したか否か）。

YESの場合，Htrueに含まれる節集合およびこの時点の真なる仮説の集まりを出力し，(4)へいく。

NOの場合，(2)へ戻る。

(4)(1)に戻る。

このように無限ループになるので，人間が判断して，正しい仮説の集まりが出てきたところでループを打ち切る。

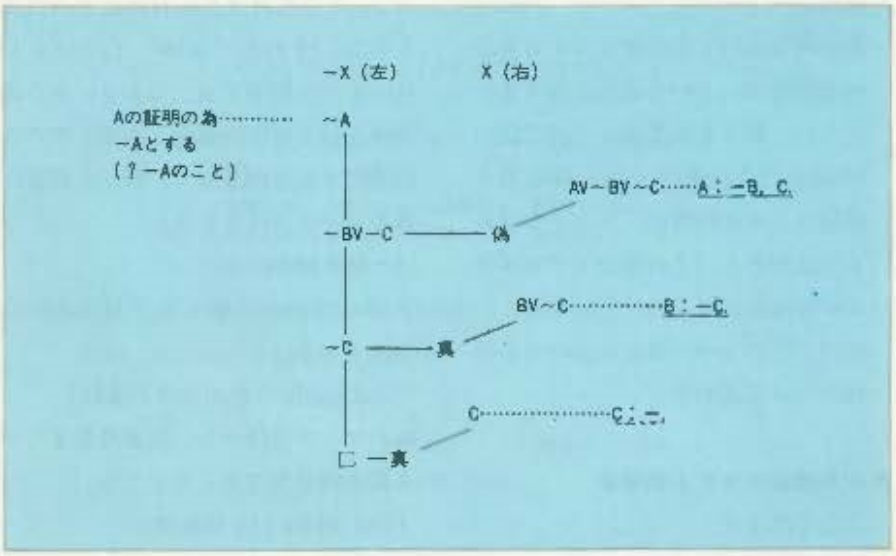

図1.2　Aを証明する導出木

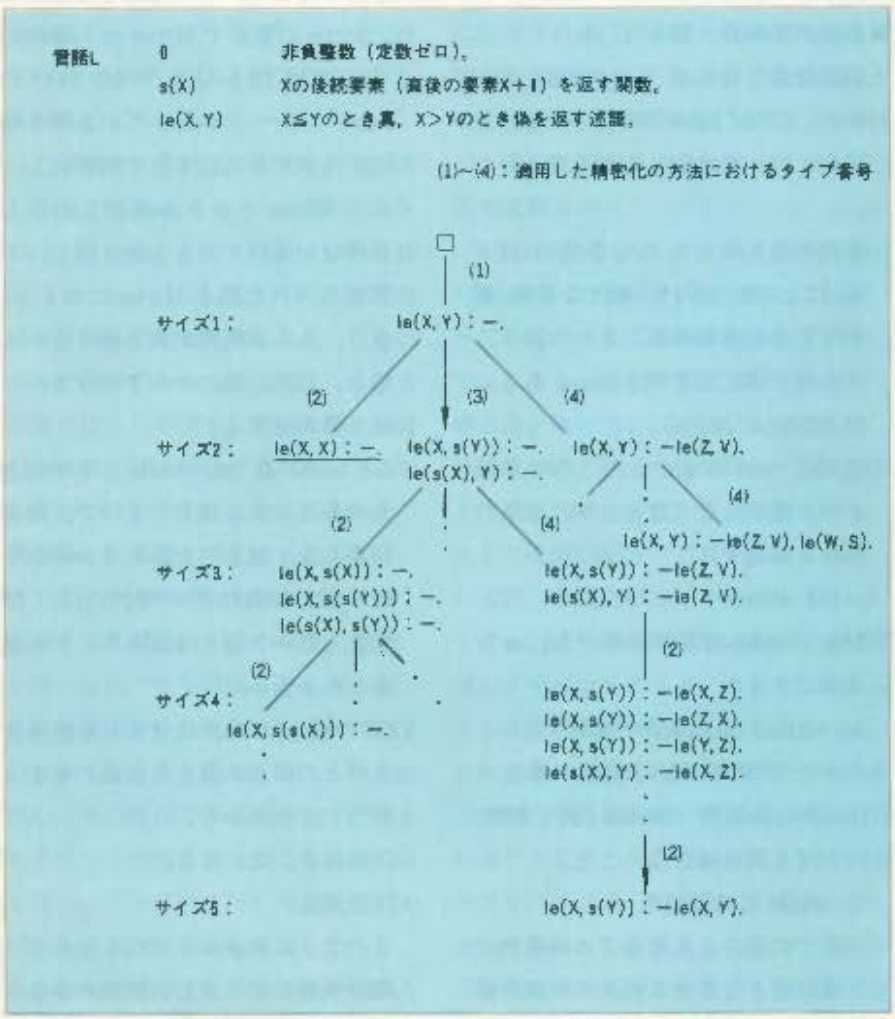

図1.3　le(X, Y)からの精密化例（仮説空間における探索木の生成例）

そしてこの正しい仮説の集まりが生成されたプログラムである。

この打切りのところをMIS自身がやらないのを疑問視する人もいると思うが，この点は，本来の自然現象の定理の帰納的獲得に類似している。

矛盾点追跡

1st while文の条件が成り立つということは，今考えているモデルの下で偽になる事例を導く偽仮説がHtrueに含まれていることを意味する。こうした仮説はその偽事例の導出証明の左成分と右成分の2分導出木を空節から逆にたどることによって求めることができ，これを命題論理の場合を例として次に説明する。

H＝｛(A：−B，C.)，
　　(B：−C.)，
　　(C：−.)｝
Strue＝｛B，C｝
Sfalse＝｛A｝

とした場合，Aを証明する導出木（Hから導出できるSfalseに属す節）は，**図1.2**のようになる（なお，∨，~は各々論理和，否定の記号で，A：−B，C，を論理式で表現すれば，A∨~B∨~Cであり，同じく，B：−C.はB∨~Cであることに注意）。

導出は背理法を使うので，~Aから空節□を導いたということは，即ち偽事例であるAを真であると導いてしまったことである。

矛盾点追跡では，最後の結果である空節□から出発し次のようにして導出木を上にたどる。

導出木の各分岐点において，導出によって消去された命題の真偽を調べ（不明な場合は人間に聞き），その命題

Xが真であれば～Xを含む枝に，偽であればXを含む枝に進む。そして，導出木の葉に到達すると，その葉に書かれているHtrue中の節を矛盾の原因と考え，Htrueより取り除く（実際にはH中の節に“false”のマークをつける）。

この例では，

□ではStrucに属するCが真なので

——→ ～Cの枝に，

～CではStrueに属するBが真なので

——→ ～B∨～Cの枝へ，

そしてSfalseに属するAが偽なので

——→ A∨～B∨～Cの葉に到着し，これに対応する規則A：－B，C．に“false”のマークを付け，Htrueより取り除く。

精密化

2nd while文の条件が成立するということは，Htrueに含まれる仮説が不足していることを表わす。そこで以前Htrueにより取り除いた仮説（“false”のマークが付いたもの）を精密化し，それをHtrueに付け加え，新たなHtrueによってStrueに含まれるすべての例が導出できるかどうか検証する。

今，精密化したい節をA，それを精密化したものをBとすると，精密化の手法としては，次のようなものがある。

タイプ(1)

A＝□ならば，B＝p（X1，…，Xn），pは言語L中のあるn引数の述語（原子論理式）である。

タイプ(2)

A＝p（X1，…，Xn）ならば，Aに登場する，ある変数Xiの全ての出現箇所にXj(i≠j)を代入したものをBとする。

タイプ(3)

A＝p（X1，…，Xn）ならば，Aに登場する，ある変数Xiの全ての出現箇所に言語L中のあるm引数の関数f（Y1，…，Ym）を代入したものをBとする。但し，m個の引数Y1，…，YmはAに現れない相異なる変数である。

タイプ(4)

A＝p（X1，…，Xn）←q（Y1，…Ym）ならば，B＝p（X1，…，Xn）←q（Y1，…，Ym），r（Z1，…，Zk）とする。r（Z1，…Zk）は言語L中のあるk引数の述語を表わす。但しZ1，…，Zkは述語p，qに現われない相異なる変数である。

精密化は1から精密化毎に1ずつ増加させかつサイズ内で行なうという制約が課されている。サイズとは，確定節のサイズ（＝その確定節に含まれる区切り記号を除いた記号の出現回数）からこの確定節に含まれる異なる変数の個数を差し引いたものを言う。

このサイズが制約サイズ内のもののみ

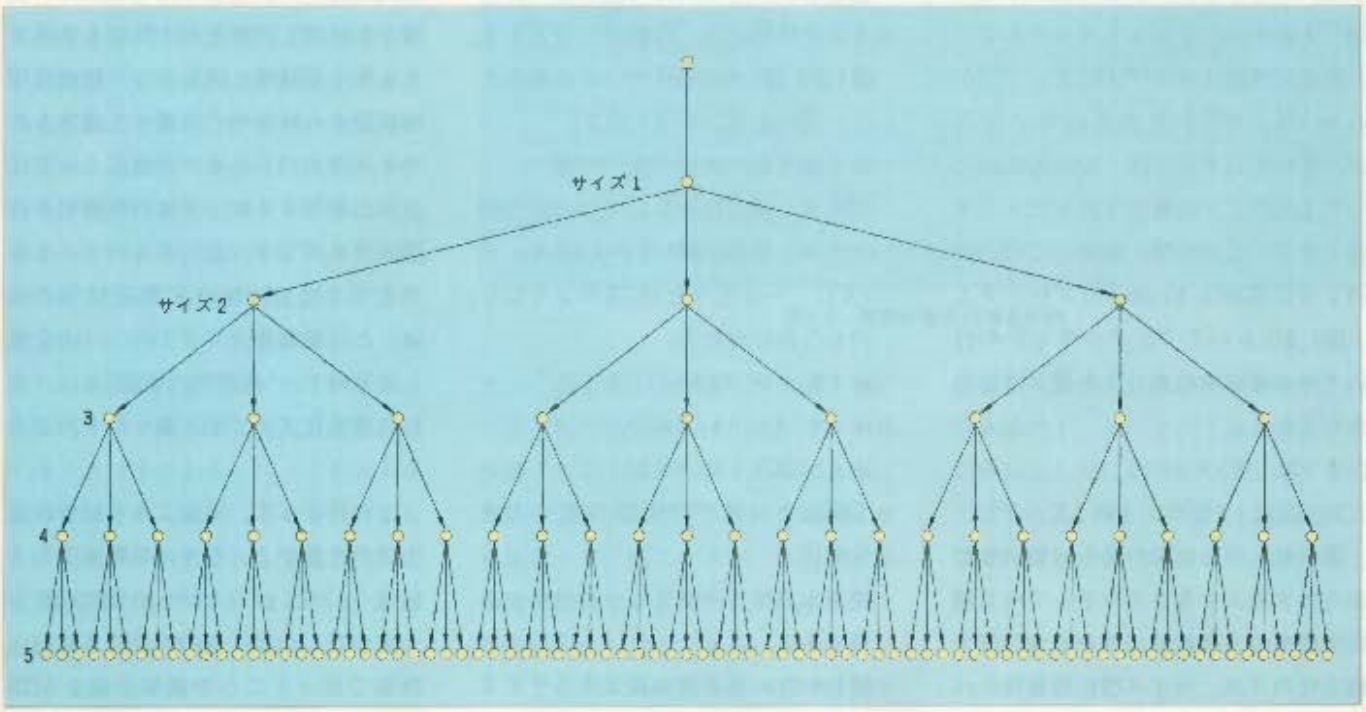

図1.4 〈精密化戦略〉探索木の分岐

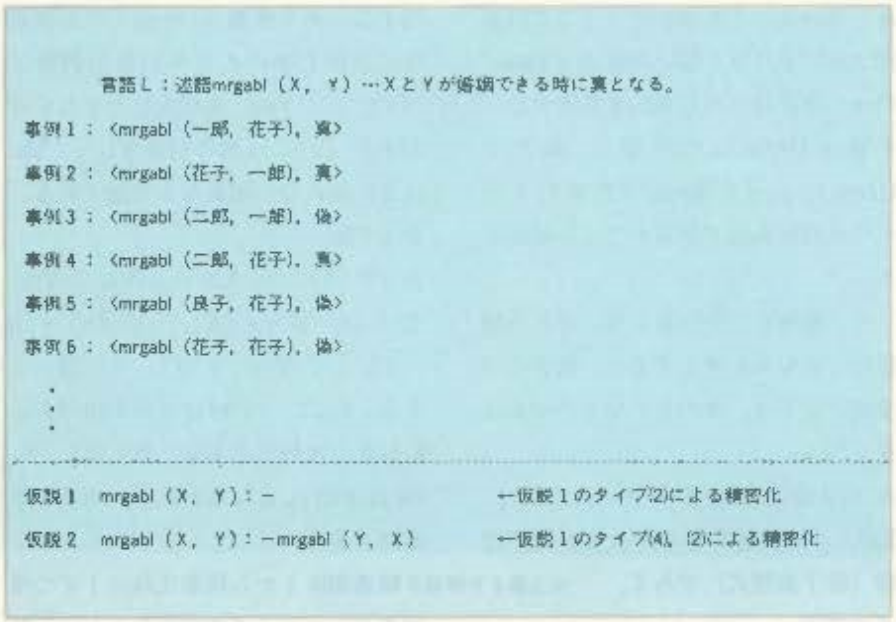

図1.5　言語が1つしかない場合の例

を仮説に加える。

図1.3にle（X，Y）からの精密化例を示す。ここで，le（X，s（Y））：－le（X，Y）：－．のサイズは，記号の数がle，X，s，Y，le，X，Y，の7個，異なる変数の個数がX，Yの2個であるから，7引く2で5である。

例えば精密化タイプ(1)によって□からle（X，Y）が生成されたとすると，それからタイプ(2)～(4)の方法によって上図のような新たな節が次々と生成される。この結果，最終的には，次のように求めるPrologのプログラム（図1.3においてアンダーラインの付いている確定節の集合）を見つけることができる。

le（X，X）：－．
le（X，s（Y））：－le（X，Y）．

図1.4は，ある仮説の最も一般的な形から逐次細大もらさず枚挙してゆく精密化戦略の，選択肢の数の増大程度を精密化の尺度，サイズ順に階層化し，仮説候補の探索木の分岐具合を図示したものである。

各選択肢の先（○印）は，図1.3にその一部が例示してある仮説例のように，ある仮説の引数の数や代入する関数の引数の数に従って生成され得る仮説数が決まるので，選択肢の数と掛け合わせると全体的には，指数的に増大する。

図1.3と図1.4で着目すべき注意点として，次の2点が挙げられる。

○原子論理式のみの枚挙の弊害
探索木の選択肢をある1つの縦方向にとった場合，図1.3ではle（X，s（X））：－がそうだが，次のようなものもこれに当たる。
le（X，s（s（X）））：－
le（s（X），s（X））：－
le（s（X），s（s（X）））：－

○Strueの検証での仮説の並べ方優先順位
探索木の横方向優先などの戦略が必要である。

図1.4では，精密化の探索木をサイズ5まで挙げてあるが，段々膨大な選択肢数になることが分かる。これは真仮説の生成を見過ごさないためには重要であるが，反面，単純モデルの場合でも仮説候補の探索枝がすぐに膨れあがりコンピュータの処理能力の限界で立往生してしまう，生成仮説があいまいな論理式ばかりで最終的にプログラムの完成品の一部に成りうる真仮説が見つからない，といった問題が生じる。

こういった問題をクリアするには，検証の回数を減らすため変な論理式は作らないようにする必要がある。

例えば，le(X，s(Y))：－le(X，Z)，におけるZは独立した変数であり，le（X，s（Y））：－le（X，Y），ではこのような独立した変数はない。このような任意的すぎる独立した変数のない論理式のみを生成するためには，サイズを逐次増加させる制限をはずす必要が出てくる。つまり精密化用の仮説は段階的精密化の進行中，ずっと保持し，これからの派生である検証用の仮説とを区別し，検証用の仮説を生成できるサイズ段階に達したら，精密化用の仮説から検証用の仮説を生成する。サイズはそのときその仮説にとっては2以上増加するが，全体の精密化の段階はサイズ1ずつ進行するので，生成のとりこぼしはない。「精密化用の仮説」とは精密化タイプ(1)，(3)，(4)を施したもので，「検証用の仮説」とは，これに精密化タイプ(2)を施したものである。

これによって，以前よりも10倍仮説生成の性能がよくなり，処理速度も3倍速く，MISから人間への質問回数も大変少なくなる。他の方法でも結局は，理論で言うところの領域知識をMISに与えることである。

MISの具体例

ここでは，MISのプログラム合成過程を分かりやすく解説するために，民法731条の婚姻適齢規定「男は満18才，女は満16才にならなければ，婚姻することはできない」を例に，これのプログラム化を考えてみる。

図1.5は，言語が一つしかない場合の例で，XやYが男か女かとか，その年齢，といった判別における具体的な言語を与えられていないため，精密化タイプ(3)は使えない。

仮説が空の初期状態で，まず真事例1が到着すると，MISは精密化タイプ(1)を用い，仮説1を生成する。精密化や仮説生成はサイズでステップ進行させ，まずはサイズ1で，あらわせるもののみを生成する。

図1.5の場合，最初の仮説は1個しかできない。これだと，1～6のどの真事例の到着があっても真となり，事例3や5の偽事例までも真としてしまう。したがって，MISは事例3が到着すると仮説1から偽事例3の検証を試み，偽と解釈できないことがわかると，検証経過を追跡して使われた仮説の中から偽仮説を指摘し除去する（今は仮説1しかないため，これが除かれる）。

次にMISは真事例の検証をしようとするが，偽でない仮説がないので失敗し，偽仮説の精密化にとりかかる。

仮説2のサイズは2であり，仮説3のサイズは4である。ヘッドやボディの中で同じ項（引数にある中味）をもつ同じ述語は消去され，生成されても仮説には加えられない（例えばmrgabl（X，Y）：－mrgabl（X，Y），がそうである）。

この精密化が終わるとMISはその生みの親である偽仮説1を消去する。つまり親の偽仮説1は死に，その子孫仮説2，3に課題を託すわけである。

この子孫仮説によると，真事例1すら仮説2では偽になるが，仮説3があるので問題ない。こうして真事例4，偽事例5にも対応できる。

ところが，偽事例6の到着で仮説2が偽仮説として除かれる。今度は再び精密化を要せず，仮説3から今までの全真事例1，2，4がうまく真となり，以降はどんなに事例がやってきても同じである。このようにして，MISは（真なる）仮説つまりプログラムを作り上げる。

図1.6は，さらに判別を具体化するための複数の言語を追加したもので，**図1.5**に比べて言語や事例の記述が数段，

```
言語4 :  述語  mrgabl（X，Y）  … XとYが婚姻できる時に真となる
         述語  ge（X，Y）      … X≧Yのとき真，X＜Yのとき偽を返す
         述語  bachelor(X)     … Xが独身男性の場合真，以外偽
         述語  spinster(X)     … Xが独身女性の場合真，以外偽
         関数  age(X)          … Xの年齢を返す
         述語  male(X)         … Xが男性の場合真，以外偽
         述語  female(X)       … Xが女性の場合真，以外偽
         述語  unm(X)          … Xが独身の場合真，以外偽
         定数  16，18          … 事例や仮説に currently
         自然数 1～99          … 事例にのみ用いる
事例1 :＜mrgabl（一郎，花子），
        male（一郎），female（花子），
        unm（一郎），unm（花子），
        ge（age(一郎)，18），ge（age(花子)，16），真＞
事例2  ＜bachelor（一郎），male（一郎），unm（一郎），真＞
事例3  ＜spinster（花子），female（花子），unm（花子），真＞
事例4  ＜spinster（一郎），偽＞
          ・
          ・
仮説1  mrgabl(X，Y)：－bachelor(X)，spinster(Y)．
仮説2  bachelor(X)  ：－ge（age(X)，18），male(X)，unm(X)．
仮説3  spinster(X)  ：－ge（age(X)，16），female(X)，unm(X)．

        仮説1の精密化過程例                          精密化タイプ
mrgabl（X，Y）：－．                          ←タイプ(1)による
mrgabl（X，Y）：－bachelor(Z)．                ←タイプ(4)による
mrgabl（X，Y）：－bachelor(X)．                ←タイプ(2)による
mrgabl（X，Y）：－bachelor(X)，spinster(Z)．    ←タイプ(4)による
mrgabl（X，Y）：－bachelor(X)，spinster(Y)．    ←タイプ(2)による

        仮説2の精密化過程例                          精密化タイプ
bachelor(X)  ：－．                           ←タイプ(1)による
bachelor(X)  ：－ge（Y，Z）．                  ←タイプ(4)による
bachelor(X)  ：－ge（age(Y)，Z）．             ←タイプ(3)による
bachelor(X)  ：－ge（age(Y)，Z）．             ←タイプ(2)による
bachelor(X)  ：－ge（age(Y)，18）．            ←タイプ(2)による
bachelor(X)  ：－ge（age(Y)，18），male(Y)，unm(Z)．  ←タイプ(4)による
bachelor(X)  ：－ge（age(Y)，18），male(X)，unm(Z)．  ←タイプ(2)による
        仮説3は仮説2と同じような過程を経る
spinster(X)  ：－，から次に至る
spinster(X)  ：－ge（age(X)，16），female(X)，unm(X)．
```

図1.6　言語が複数の場合の例

面倒になっているが，これで理論的にはMISより，婚姻適齢規定の正しいプログラムが生成できる。

MISのこれから

MISは原理としては，夢を実現しているが，事例の与え方が問題（順序や量の面）で，苦労して事例の与え方を工夫しても，大したものはできない。MISには学習の広範囲な展開上，障害となる点が少なくとも2つある。一つは，教師（オクラル）に対して際限なく知識（＝事例）を要求することであり，もう一つは，新たな述語をシステムみずからが生成していく膨大な自由度を放棄したことである。

これらの問題の解決も含めて，MISでは次のようなアプローチがなされている。

TMSの利用

DoyleのTMSは，非単調論理を実現した人工知能向けデータベース管理システムで，その主要な機能は，新しい知識がデータベースに付け加わったときに既存の知識との整合性を保つTruth Maintenance（TM），および矛盾が発見された際に仮説の集合を見直すことにより矛盾の解消を行なうDependency Directed Backtracking（DDB）の二つである。

この拡張版のATMS（Assumption-based TMS）を用いてMISにオクラルからの知識（事例）の誤りが混入しても，そちらをうまく排除させることに成功させた例が発表されている。

HMISの提案

MISの推論手続きは，論理的基盤に基づいて定式化されており，帰納推論メカニズムを演繹推論の枠組みで整理するのに成功したものと見なすことができる。論理による定式化は，モデル推論メカニズムを形式的に調整することに貢献する反面，自然現象の観測データに本質的に内在するノイズへの許容性がないという欠点をもつ。つまり，オクラル（神託＝人間）の与える事例列中には真／偽値のウソはないという前提に立つという制限がある。

そこで，オクラル集合を説明するモデルを構築できないときに，その原因となるオクラルを発見し，矛盾を解消する方法として仮説型モデル推論システムがある。

仮説型モデル推論システム（HMIS＝Hypothetical MIS）はATMSを用いることによって，オクラル集合の変化に際して，過去の推論結果から，矛盾なく利用可能な推論結果を選別することができる。

ロンドンでの成果

英国Turing研究所からはCIGOL（LOGICの逆）という名のシステムが発表されている。

MISでは，言語は与えられるもので，得られる仮説も用いる言語の範囲になっているが，CIGOLシステムでは帰納推論の過程で新しい述語（つまり言語）を“発明”できることが大きな特徴である。また，与えるべき事例列の数もMISに比べ驚くほど少なくてすむようになっている。

正事例（真事例）の活用および汎化の理論

MISでは，正事例（真事例）が枚挙された論理プログラムの正当性のチェックにしか利用されず，推論を実際に進めるのは負事例（偽事例）である。そこで正事例をより有効に推論に生かすために，Plotkinによって与えられた最小汎化（least generalization）を利用する推論手法が提案されている。また，今話題のニューラルネットとの組合せなども有望視されている。

㈱インターナショナルソフトウェア（ISC）では，IBM5550（PS/55）シリーズやNEC・PC-9801シリーズなどのMS-DOSで動くMISを，簡単なProlog処理系（名称：CMIS）とともに頒布しています。

媒体　5インチ2DD／2HC／2HD，
　　3.5インチ2HD
価格（税込）　実行形式：1万円，
　　ソース：5万円

問合せ先

〒112 東京都文京区小石川1-15-17
㈱インターナショナルソフトウェア
TEL 03-817-0011
研究開発部（担当：中村）