発見的プログラミング

辻　　三　郎*

1. まえがき

人工知能 (artificial intelligence) の研究は，情報科学の主要課題のひとつとして，最近多くの人々の関心を集めている．“知能”という言葉自体が，かなりばくぜんとした定義しか与えられていないことから考えても，人工知能と言う名前で総括される研究分野も，自己構成系，学習，感覚システム，問題解決，言語，巨大な知識のデータベースなど広範囲にわたり，しかもいろいろの方向からのアプローチがなされているのも当然であろう．この中で終始ひとつの中心課題となっているのが，ディジタル計算機による問題解決 (problem solving) である．

今までの情報処理の姿は，与えられた課題をまず人間が，それを解くアルゴリズムを考えてプログラム言語に書き下し，コンピュータはそれにしたがって論理演算を行なうに過ぎなかった．もしも機械が，人間のように与えられた問題を解く知的能力を持っていれば，自然言語で書かれた問題を与えれば，コンピュータが独力でその問題の解法を発見し，その結果を人間に報告することも可能になろう．もちろん，現状ではこの目標はあまりに遠く，今後多くの厚い壁を乗り越えねば達成されないことは言うまでもない．しかしながら，幾つかの限定された局面に対して，コンピュータが問題を解決する道すじを見出す研究は，着々となされている．たとえば，コンピュータによるゲームやパズルの解法，数学の定理の証明，時刻表の作成や検査結果からの化学式の推定，質問に回答する系，ロボットの仕事の計画などがあげられる．これらは，一見すると互に関連がなさそうに感じられるが，実はいずれも上に述べた問題を解く能力を機械に与える研究の一環であり，これらに共通して発見法 (heuristic) の考えが用いられている．

問題解決の研究では，研究者は工夫した発見法をコンピュータプログラムに書き下し，それによってコンピュータが与えられた問題を有効に解くことができるか実験する．この過程を通じて，人工知能に本質的な発見法の姿を明らかにしようとするわけである．本稿においては，発見法を用いたコンピュータプログラム，すなわち発見的プログラミング (heuristic programming) に共通する考え方や手法，さらにその応用などを，この分野にはあまりなじみのない読者を対象にして解説しよう．なお最近すぐれた単行本[1~6]も出版されているので，興味のある読者は参照されることを希望する．

* 大阪大学基礎工学部

2. 発見法とアルゴリズム

発見法についても教育学や情報科学の立場から，いろいろの定義がされているが[7]，ここでは具体的な問題を通じて，その性格を考えてみよう．人工知能の歴史は，1949年の Shannon の論文から始まったと考えても差支えなかろう．彼は，コンピュータがチェスを指すプログラムを作る課題に含まれている本質的問題点を明らかにした．すなわち，チェスは有限のゲームであり，ある局面に対して有限個の指し手しかない．したがって，与えられた局面から出発して自分と相手方が指しうるすべての手をたどって行くと，必らず終局（勝ち，負け，または引き分け）の状態になる．結局，現在の局面から相互の指し手によって枝分かれする有限の木（tree）としてチェスは表現でき，この木の全体を高速のコンピュータでたどれば，現局面での最良の指し手は簡単に決定できるはずである．

このような考え方はチェスだけでなく，碁，将棋，チェッカーなどのゲームに共通して適用できる．しかし問題点は，それぞれのゲームに対応する木の大きさである．この大きさは，ゲームの複雑さに従って急速に天文学的数字に近づく．チェッカーは，チェスに比べるとはるかに簡単なゲームであるが，それでもその木にある節点の数は 10^{40} 程度と概算される．今かりに1ナノ秒当り3個の節点を調べられる超高速コンピュータがあったとしても，すべての節点をチェックするには 10^{21} 世紀の年月が必要である．チェスの場合の節点数が 10^{120} にも達することを考えると，このよう

にゲームの木をすべて調べて，論理的に最善手を決定することは不可能であると結論されよう．

そこで Shannon は，このようなやり方ではなく，与えられた局面に対して幾つかの限られた手を読んで，その結果の局面を比較検討して，考えた手の中でよさそうな最初の一手をえらぶことが必要であると示唆した．われわれ人間も，碁・将棋などの知的ゲームにおいて指し手を決定する時には，似たような方法がとられていることから，このやり方は人間の知的活動のひとつの型となっていると類推できる．

チェスのプログラムは，発見的プログラムのひとつの典型的例と言えよう．すなわち，与えられた問題を完全に解くアルゴリズムは，コンピュータの容量や計算時間の制限を除けば，(すなわち数学的に考えれば) たしかに存在する．しかし，われわれに許された条件下では，(すなわち工学的に考えれば) それはあまり意味がなく，もっと有効に解をみつけるやり方が重要である．今までに研究されたチェスのプログラムは，この制約条件を考慮して作成され，コンピュータが現局面でよさそうな手を発見するやり方に，いろいろな工夫がされている．現在の所では，コンピュータのチェスを指す腕前はかなりのものであるが，すべての人間に対して勝つところにはなっていない．

ここで，問題を解く時のアルゴリズム (algorithm) と発見法の差異を考えよう．アルゴリズムを見出すには，与えられた問題の持っている特徴や性質から，それを解く手続きを与える．その解法がアルゴリズムと呼ばれるには，解く手続きを続ければおそかれはやかれ問題の解を求められるという保証が必要である．もちろん，上述したゲームの木をすべて調べてチェスの最善手を決定するアルゴリズムのように，多くの費用や時間が必要となり非実用的になる場合も数多く存在する．

一方，発見法は複雑な問題を解く時の能率を良くするための経験法則，方策と定義できる．そして，その問題を解くかもしれないが，必ずしも解けるという保証はされていない．チェスの場合には，最善手を指すことは少いとも言えよう．もっともこれは，人間の指し手も同様である．

さて一般的に，われわれ人間が与えられた問題を解く過程を考えると，発見法的なやり方，問題の性質や過去の経験を考えながら解く道すじをいろいろと探索していることに気がつく．与えられた問題によっては，最終的にはそれを解くアルゴリズムを見つけ，また多くの複雑な問題ではアルゴリズムと発見法を併用することもある．しかし，そこにたどりつく過程では何らかの発見的手法が用いられていることから，コンピュータにこのような知的能力を持たせることが，人工知能を達成する条件のひとつと結論されよう．

3. 問題の表現

コンピュータによる問題解決を行うには，まず問題を計算機内で取り扱いやすいように表現することが必要である．真の人工知能と呼ばれるためには，普通の人間に問題を与えるように自然言語で問題を記述し，計算機がその問題の意味をほん訳して，その解決法を探すべきであるが，現在の発見的プログラミングの研究では，多かれ少なかれ人間が手助けしてその問題の本質的部分を計算機向けに表現し，それを対象にコンピュータが解を探索する部分に研究の重点がおかれていると思われる．事実問題の表現の仕方によって，同じ問題でもそれを解くむつかしさは大いに影響される[8]．

グラフによる表現

問題解決の過程は，普通グラフの形で記述される．わかりやすい例として，人工知能の研究でしばしば現われるハノイの塔のパズルを取り扱ってみよう[2),3),6)]．このパズルは，図1に示した3本の棒のひとつ棒1にささえられた大きさの異なる円板 A, B, C のすべてを棒3に移す問題である．円板を移動させるのは，必らず1個ずつに限られ，しかも小さな円板の上により大きな円板をおいてはいけない．

さて，このパズルの各局面は，それぞれの棒にどの円板がおかれているかという文字列で表現できる．たとえば (A, B, C) (φ) (φ) は最初与えられた局面で，目標は (φ) (φ) (A, B, C) で示される．次に許された移動を，ある状態から次の状態に移動するオペレータと考えよう．一般化すると与えられた問題は，初期状態の集合 $S=\{s_1, s_2, \cdots, s_m\}$，オペレータの集合 $F=\{f_1, f_2, \cdots, f_r\}$，目標状態の集合 $G\{g_1, g_2, \cdots, g_n\}$ によって (S, F, G) の形で記述される．

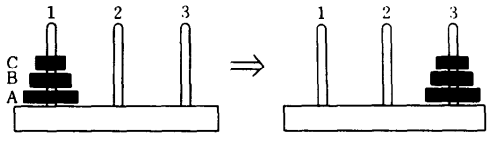

Fig. 1 The Tower-of-Hanoi Puzzle

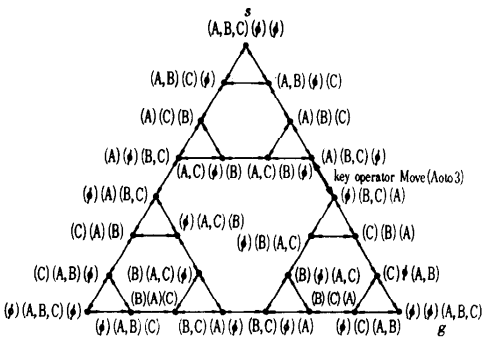

Fig. 2 State Space Graph of the Tower-of-Hanoi Puzzle

さてハノイの塔の問題では，Sに対するオペレータは Move (C to 3) と Move (C to 2) で，これによって2個の状態が生成される．それらについて次々とオペレータを操作して生成できるすべての状態によって作られる空間を，このパルズの状態空間 (State Space) と考える．この状態空間は，図2に示すグラフの形に書き直すことができる．グラフのノードはパズルの各局面（状態）で，アークはオペレータである．問題の正解は，ノードSとGを結ぶ経路として得られることは明らかである．しかし前節で述べたように，たいていの問題ではグラフ内のノード数は天文学的数字となり，その探索に発見的プログラミングが必要となる．

ここで示した例では，状態の表現に簡単な文字列を用いているが，一般的には問題の構造・属性を表現するデータ構造が用いられる．

AND/OR グラフ

チェスのような2人で行なうゲームは，パズルに比べてより一般的な AND/OR グラフで表現できる．図3に示すように与えられた局面Sから出発して，交互にさす手によって作られるグラフ内を探索して，勝ちの局面Gに至る経路を見つけるところは同じであるが，各ノードにおける選択の条件は異なっている．現在の局面が必勝であると証明する（すなわち勝つ手を見つける）には，自分の手番Sで生成される局面 n_1, n_2 のどちらか (OR) が必勝であると判ればよい．相手の手番 n_1 では，そこで生成される局面 n_{11}, n_{12}, および (AND) n_{13} のいずれもが必勝であることが，勝ちの条件である．このように2人ゲームの問題は，OR と AND のノードが交互に現われるグラフ内の探索問題に変換できる．一般的にある与えられた問題を解くには，それをより簡単な問題に分解し，それら

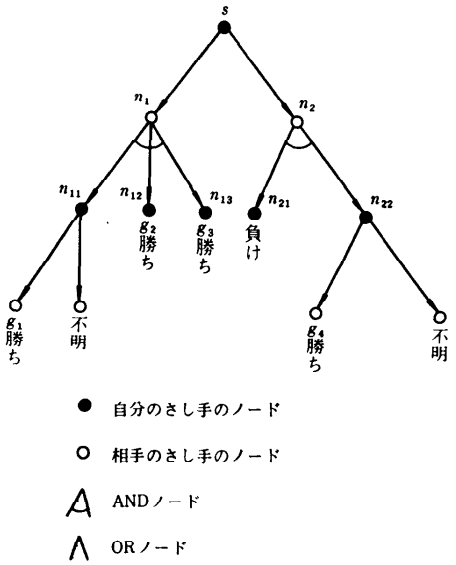

Fig. 3 AND/OR Tree of Two Person Game.

の AND/OR グラフの形に書き直す手法がとられるので，AND/OR グラフの探索が問題解決の基本型となっている．なお最初に述べたパズルは，OR ノードばかりで作られる簡単化された場合と考えられる．

述語論理による表現

今まで述べたパズルやゲームの問題は，ある限られた狭い対象を取り扱っており，その解法も有効な探索に主体がおかれてきた．もっと一般的な問題を解くには，その問題の構成をより自由に記述し，それに基づいて論理的推論を行ないながら解を探索することが必要である．すなわち，我々人間と同様の常識をコンピュータが持っていて，自己の知識に基づいて与えられた問題を解決する[9),10)]ことが遠い将来の目標であり，そのためにコンピュータの巨大なデータ・ベースにこの世の中の法則や事実を記述しておき，それを用いて推論を行なうシステムの研究が行なわれている．

問題の記述には，最初数学の定理の証明などに用いられていた述語論理 (Predicate Calculus)[11),12)] が使われ，各種の探索法により論理的推論を行なっている．

4. 探索法

4.1 状態空間内の探索

グラフ生成手続き

開始ノードsとオペレータFが与えられて，目標G

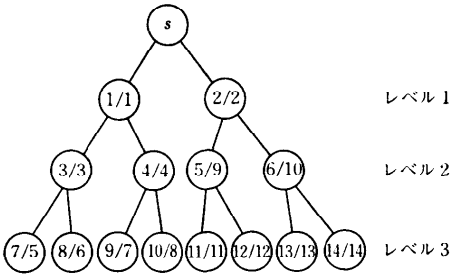

Fig. 4 Bread First and Depth First Procedures

に達するグラフを生成する方法の基本型は，盲目的探索法(Blind Search Method)である．これは，sから機械的に接続するノードを作り出すもので，横型(bread first)プロセスと縦型(depth first)プロセスがある．図4に示すように，横型ではsより距離1のノードをすべて生成し，ついで距離2，距離3のノードを順次すべて生成していく．一方，縦型プロセスでは，あるノードに直接つながっているノードはすべて作るが，次はその中で一番左のノードについてのみ生成を行う．このようにして，あらかじめ定められた深さ（図4ではレベル3）まで達しても目標が見つからない時には，今まで生成されたノードのうち最も新しく生成されたノードから再び生成をくり返す．いずれの方法もグラフ内のノード数が多くなると，解を発見することは困難であるが，発見的手法を用いる時と比べると各ノード当りの計算時間が短く，無駄な探索を省くなどの工夫をこらすと，発見的プログラミングに匹敵することもある[13]．

MINI-MAX 手続き[14]

何度も述べたように，グラフ内のノード数が多くなると盲目的探索法では処理しきれなくなり，何らかの発見的手法が必要となる．すなわち開始ノードから出発して比較的少数の局面を調べ，その中で最も有望な1分岐を選択する．局面の有望性を比較するために，評価関数が導入されるのが普通である．たとえば，チェッカの場合は，キングの数の差m，平駒の差n，自由に駒を動かせる場所の差uで局面の優劣の比較ができる．たとえば，評価関数をこれらの一次結合として $ak+bm+cu$ $(a, b, c$ は定数$)$ と定めれば，生成された各方面の点数が算出できる．現局面sから出発して，4手先きまで読みその結果着手を決めることにしよう．図5はsから生成されるAND/OR木で，レベル3の各ノードの数字は局面の評価関数の値である．さてレベル3の局面の評価によって，レベル2における手が決定される．ここは自分の手番であるから，最大の評価が得られる枝（図5で太線で示す）がえらばれ，その値がレベル2の評価値としてくり上げられるので，マックス局面と呼ばれる．レベル1は，相手側の手番で逆に最小の評価値をもつノードにつながる枝がえらばれる．このように，ノードがANDかORかによって，下位局面の評価値の最小または最大値が順次くりあげられ，最後にsでの着手が決定されるので，ミニマックス手続き(minimax procedure)と呼ばれる．

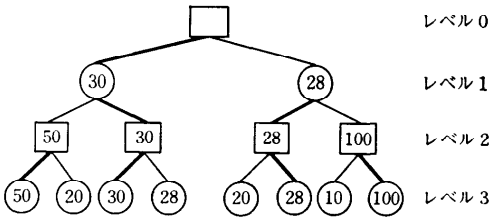

Fig. 5 Minimax Backing-Up Procedure

ミニマックス手続きを改良した(m, n)手続きは，数個のより良い後続局面を持つ枝が，ただひとつの良い局面を持つ枝より有利と考え，マックス局面ミニ局面のそれぞれについて，上位m個または下位n個の評価値をもとにして手の選択を行っている[15]．

α-β 手続き

ミニマックス手続きも，評価する局面のレベルが深くなると，大きな木の探索問題となり計算時間が問題になる．縦型ミニマックス手続きと等価な結果が得られ，しかも探索するノード数をかなり節約する方法が，$\alpha\beta$ 手続き ($\alpha\beta$ procedure) である．図6に示すAND/OR木では，レベル6の評価値から出発して手の決定を行なっている．縦型ミニマックス手続きでは A_6, B_6 の評価値から $A_5=\mathrm{MIN}(A_6, B_6)=0$. 次に C_6 の評価値から $B_5 \leqq C_6=-3$. したがって，$A_4=\mathrm{MAX}(A_5, B_5)=0$. このことから，$C_6 \leqq A_5$ から直ちに (D_6 の評価は不要) A_4 の値が決定できる．A_5 の値は A_4 のマックスに保証されている最小値でαと呼ばれる．もしもマックス局面にくり上げられる値がαをこえぬことが判れば，α カットをおこし今評価したマックス局面 (C_6) の前局面 (B_5) で生成し残したノード (D_6)

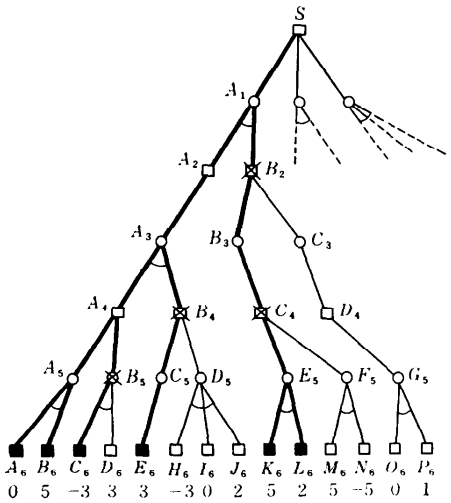

Fig. 6 An example Illustrating the Alpha-Beta Procedure

は捨て去り，前々局面 (A_4) で残っている後続局面を検討する．一方，ミニ局面に保証されている最大値 β は，A_4 の値から0に設定される．$C_5=E_6=3$ が β より大であるから β カットをおこし，D_5 とその下位ノード F_6 より H_6 の評価が不要となる．α, β の値は，新しい値が発見されるたびに更新される．

$\alpha\beta$ 手続きが有効であるには，比較的早い探索の時期でよい手を見つけて，多くの α カット，β カットをおこすことが必要である．最適の場合には，深さ D レベルまで評価する $\alpha\beta$ 手続きの評価するノード数と，$D/2$ レベルまで評価するミニマックス手続きのノード数とが同程度になる．

$\alpha\beta$ 手続きの能率を向上させるため，開始ノードの近傍の浅いレベルを探索し，その評価値の高い順に後続局面を並べかえる固定順序づけ (Fixed Ordering) の手法が用いられる．最初の並べかえの後，深い探索を行ない，より正しい情報に基づいて次々と並べかえを行なう動的順序づけ (Dynamic Ordering) によりさらに好結果が得られる[15]．

多目的発見プログラム MULTIPLE は，どのノードから発芽するかを，前提が正しい確率と価値関数の評価によって定め，$\alpha\beta$ 手続きに比べて良好な結果を得ている[5]．

4.2 GPS

Newell, Shaw, Simon によって実験された GPS (General Problem Solver 一般問題解決プログラム) は，今まで述べてきた個別的に問題を解くプログラムと異なり，適当な言語で記述されていれば数学定理の証明，不定積分，各種のクイズなど広範囲の問題を解決することができた[1),2)]．GPS の解法は，与えられた問題をよりやさしい問題の組合せに変換し，個々のやさしい問題を解くことによることにより原問題を解決するやり方に等価で，問題置換探索法 (Problem Reduction Search Method) とも考えられる[6]．

図1のハノイの塔のパズルを例に説明すると，初期状態 $(A, B, C)(\phi)(\phi)$ と目標 $(\phi)(\phi)(A, B, C)$ の状態を比較する．3種の円板がいずれも棒3にない点が異なっているが，その中で円板Aが最も重要な要素と考えられる．そこで Move (A to 3) がこの問題を解決するオペレータ (Key operator) としてえらばれる．しかし，このオペレータを実行するためには，まず (A) (B, C) (ϕ) を達成しないといけない．そこで，(A) (B, C) (ϕ) を目標として，その Key operator を決定し，(A, B), (ϕ), (C) が次の目標となる．Sからこのノードへ移る方法は自明で，(A) (B, C) (ϕ) が実現できる．次に，ここを開始ノードと考えて，目標との差を同じ手法で縮少して，問題を解決できる．図7は，GPS の問題解決のブロック図を示す．図1をより複雑にした4枝の円板よりなるハノイの塔の問題を，GPS は46個の副目標を作って解いた．

4.3 推論による問題解決

定理の証明と導出原理

コンピュータに推論を持たせて問題解決を行わせる研究は，Newell らの Logic Theory Machine[16] による数学定理の証明に始まった．これは，数学公理や定理を命題論理 (Propositional Logic) で表現し，公

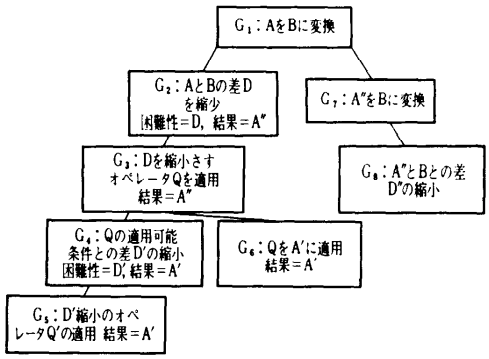

Fig. 7 Typical Goal Tree of GPS[2)]

理系から論理的組合せで得られる定理を導こうとするものであり，その後能率の良い探索法が研究された[17]．

命題論理を一般化した述語論理[11]を用いることにより，さらに広範囲の対象が取り扱うことが可能になった．今までの研究の重点は，一階の述語論理[12]で記述された公理系の証明であるが，Robinson の導出原理 (Resolution Principle) が，その解決に強力な武器となった．述語論理の半決定可能性 (Semidecidability) のため，有限の手続きで証明可能なのは定理が正しい場合に限られている．一方，より一般的な多階の述語論理で記述される論理の証明の研究も行なわれている[19),20)]．

質問回答システム

数学の論理の証明は，与えられた公理系から定理の真偽の判定を行なうものであるが，見方をかえると QA システム (Question Answering System 質問回答システム) にそのまま利用することができる[21)]．この場合，公理に相当するものがデータベースに蓄えられた事実・法則などで，証明しようとする定理がシステムに対する質問である．いま簡単な例によって，導出原理による解法を示そう．「1) 太郎が行く所に花子はついて行く．2) いま太郎は事務所にいる．」という記述がデータベースに入っているとしよう．システムに対し「花子はいまどこにいますか」という質問に対する答えを発見する過程を考える．上述した記述は，WFF (well formed formula) の形で書かれていると仮定する．すなわち

$$(\forall x)\{\mathrm{AT(Taro},x)\Rightarrow\mathrm{AT(Hanako},x)\},\quad(1)$$

$$\mathrm{AT(Taro,\ office)}\quad(2)$$

で，(1) すべての場所 x に対して，太郎が x にいることが真であれば，花子が場所 x にいることも真である，(2) 太郎が事務所にいることが真である，とほん訳される．次に質問は，花子が x にいることが真となる x が存在するか (花子が x にいるか) という形で

$$(\exists x)\ \mathrm{AT(Hanako},x)\quad(3)$$

で表現される．(1), (2) を公理，(3) を仮説と考えて，導出原理で証明しよう．

(1) を標準形に変換すると，

$$(\forall x)(\sim\mathrm{AT(Taro},x)\vee\mathrm{AT(Hanako},x))\quad(4)$$

導出原理では，まず仮説 (3) の否定

$$(\forall x)(\sim\mathrm{AT(Hanako},x))\quad(5)$$

と，公理の各項を比較し，Q(a) と ～Q(a) の形式の項を消去し新しい式に書き直す．この場合は，図8に示すように，最後に NIL が得られ仮説が真と証明される．

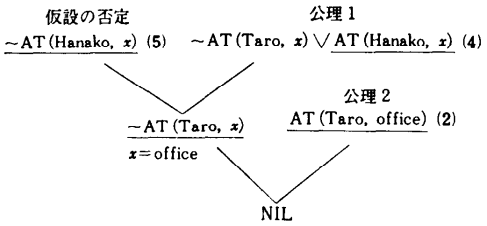

Fig. 8 Refutation Tree for Simple Example

QA システムでは，仮説の真偽のみでなく，具体的な事実が必要である．その時は，仮説をトートロジー (W∨～W) の形に書いて，証明と同様の手続きを行なう．図9は，その実行結果で第2段で ～AT(Taro, x) と AT(Taro, office) を消去する時，変数 x に定数 office が代入されるが，それは同時に AT(Hanako, x) にも代入され，AT(Hanako, office) という記述が得られる．ここで示した例は非常に簡単な場合であるが，多くの必ずしも直接関連を持たない事実を格納しておけば，与えられた質問に対して，コンピュータがこれらの事実に基づく推論を行なって回答するシステムが，原理的に可能のことが判る．もちろん対象が複雑になるに従い，定理の証明を行なうまでの手続きが長くなるので，その探索の能率向上の研究が行われている．

ロボットの計画

最近研究されている知能ロボット[22)]は，与えられた仕事に対する計画を独立でたて，それに従って目や触覚などで環境を調べながら，手足を用いて仕事を実行している．この計画をするプロセスで，QA システムの手法を用いることができる．

計画作成には，ロボットの周囲環境や法則が必要であるが，これらは world model として述語論理の形で記述される．これらは公理に相当し，仕事の目標が

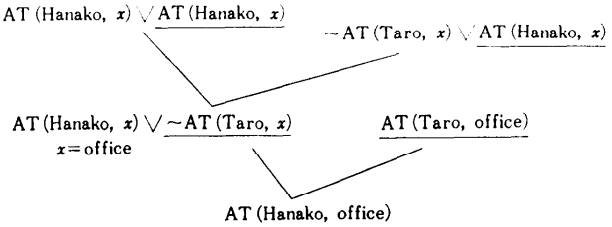

Fig. 9 Modified Proof for Question Answering System

仮説となり，導出原理により解を得ることができる[6]．ここでは，GPS の手法によるロボットの問題解決の手法を述べよう[24),25)]．

3個の箱 1, 2, 3 が b, c, d に置いてあり，ロボットが a にいる時，これらの箱を一箇所に集める計画をたてる．world model M は，初期状態では M_0 で

ATR(a), AT(BOX 1, b),
AT(BOX 2, c), AT(BOX 3, d)

で示され，最初の目標 G_0 は

$(\exists x)$[AT(BOX 1, x)∧AT(BOX 2, x)∧
AT(BOX 3, x)]

で示される．ロボットの動作が，この場合はオペレータで push と goto がある．これを実行するには前提条件が必要で，またその結果 M は書直される．たとえば，k を m から n まで押すという行動 push(k, m, n) は，前提条件として ATR(m)∧AT(k, m) が必要で，実行すると ATR(m), AT(k, m) を M より消去し，ATR(n) と AT(k, n) が M に追加になる．

解法は GPS のやり方に基づく．すなわち，G_0 から出発して探索した木が現在 $(M, (G_i, G_{i-1}, \cdots, G_0))$ となったとする．次に $(M \Rightarrow G_i)$ の証明を導出原理で試み可能の時はその結果得られたモデル M' を用いて，$(M', (G_{i-1}, \cdots, G_0))$ の証明を行なう．証明不能の時は，M と G_i の差を著るしく減少させるオペレータを探し，現在満足されない前提条件を新しい副目標 G_{i+1} として解の探索を続ける．上に述べた例では，M_0 と G_0 の差から push(BOX 2, m, b) を実行さす条件 AT(BOX 2, m)∧ATR(m) が G_1，そのために goto(m, c) が必要となり ATR(m) が G_2 となる．これは容易に実行できるので，M_0 が M_1 に変換される．以下 G_4 まで作成し，goto(a, c)，push(BOX 2, c, b)，goto(b, d)，push(BOX 3, d, b) の行動計画ができる．さらに計画を一般化して記憶する機能が追加され，似た型の仕事の計画は過去の経験に基づいて行われるようになった[24)]．

5. むすび

問題解決に用いる発見的プログラミングの初歩的部分を概説した．紙数の関係でふれられなかったが，研究の初期段階から学習方式が着目され，多くの研究がなされてきた[14),24)~26)]．またアセンブリングライン，不定積分[1)]，組合せ問題[7)]，化学式[27)]，ロボットの手の制御[28)]などへの応用も，文献を参照されたい．最後に，目標に到達する procedure を克明に記述する問題解決[29)]の手法が，導出原理の一様な探索の欠点を克服するため研究されていることを付記したい．

参考文献

1) ファイゲンバウムほか(阿部訳)：コンピュータと思考，好学社 (1969)．
2) G. Ernst & A. Newell: GPS; A Case Study in Generality and Prc'lem Solving, Academic Press, New York (1969).
3) R. Banerji: Theory of Problem Solving; An Approach to Artificial Intelligence, Elsevier, New York (1969).
4) M. Minsky, et al.: Semantic Information Processing, MIT Press, Cambridge (1968).
5) スレイグル(南雲ほか訳)：人工知能，発見的プログラミング，産業図書 (1972)．
6) ニルソン(合田，増田訳)：人工知能，コロナ社 (1973)．
7) 中村，織田：Heuristic と制御について，計測と制御，Vol. 8, No. 6, pp. 388～400 (1969)．
8) S. Amarel: On Representation of Problems of Reasoning about Actions, in D. Michie (ed.), Machine Intelligence 3, pp. 131～171, Elsevier, New York (1968).
9) J. McCarthy: Program with Common Sense, 文献 4, pp. 403～410.
10) J. McCarthy: Situation, Actions and Causual Laws, 同上, pp. 410～418.
11) 彌永ほか編：数学辞典，p. 654，岩波 (1960)．
12) J. A. Robinson: An Overview of Mechanical Theorm Proving, in R. Banerji & M. Mesarovic (ed.), Theoretical Approaches to Non-Numeric Problem Solving, pp. 2～20, Spring-Verlag, New York (1970).
13) L. Siklóssy, et al.: Breadth-First Search: Some Surprising Results, Artificial Intelligence, Vol. 4, No. 1, pp. 1～27 (1973).
14) サミュエル：チェッカー・ゲームによる機械学習の諸研究，文献 1，pp. 102～144.
15) J. Slagle & J. Dixon: Experiments with Some Programs That Search Game Trees, J. ACM, Vol. 16, No. 2, pp. 189～207 (1969).
16) ニューウェルほか：Lögic Theory Machine を使った実証的研究，文献 1，pp. 147～179.
17) H. Wang: Towards Mechanical Mathematics, IBM J. Res. Develop. Vol. 4, pp. 2～22 (1960).
18) J. A. Robinson: A Machine Oriented Logic Based on the Resolution Principle, J. ACM, Vol. 12, No. 1, pp. 23～41 (1965).
19) J. A. Robinson: Mechanizing Higher Order Logic, in B. Meltzer & D. Michie (eds.) Machine Intelligence 4, pp. 151～170, Elsvier, New York (1969).
20) J. McCarthy & P. Hayes: Some Philosophical

Problems from the Standpoints of Artificial Intelligence, ibid., pp. 463～509.
21) C. Green: Theorem Proving by Resolution as a Basis for Question Answering Systems, ibid., pp. 183～205.
22) 辻: ロボットの情報処理, 情報処理, Vol. 11, No. 8, pp. 467～475 (1970).
23) R. E. Fikes: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, Artification Intelligence, Vol. 2, No. 314, pp. 189～208.
24) R. E. Fikes, et al.: Learning and Excuting Generalized Robot Plans, ibid., Vol. 3, No. 4, pp. 251～288 (1972).
25) A. Samuel: Some Studies in Machine Learning Using the Game of Checkers II Recent Progress, IBM J. Res. Develop. Vol. 11, No. 6, p. 601 (1967).
26) D. Waterman: Generalization Learning Techniques for Automating the Learning of Heuristics, Artificial Intelligence, Vol. 1, No. 1/2, pp. 121～170 (1970).
27) E. Feigenbaum, et al.: Generality and Problem Solving, in B. Meltzer & D. Michie (eds.) Machine Intelligence 6, Elsevier, New York (1971).
28) D. Whitney: State Space Model of Remote Manipulation Tasks, Proc. Intern. Joint. Conf. Artificial Intelligence, pp. 495～507 (1969).
29) T. Wignorad: Understanding Natural Language, Edinburgh Univ. Press (1972).

(昭和48年7月12日受付)