ICOT Technical Memorandum: TM-0289

TM-0289

Efficient Search Strategies for Hypothetical Reasoning (Research Memo)

井上克己

March, 1987

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

February 19, 1987

Efficient Search Strategies for Hypothetical Reasoning

(Research Memo)

Katsumi Inoue

Fifth Laboratory, ICOT Research Center.

1. Introduction

1.1 仮説推論とは? (What)

競合する知識、あいまいな知識等を用いて推論を進める場合、それらの知識をどのように扱うかが問題となる。 その一つの手段として、推論中に仮説を立ててそれを処理することが考えられる。 そのような状況は、対象世界の知識(規則、事実、制約等)、及び問題解決のための戦略知識のいずれを扱う場合においても起こり得る。 ここで仮説となり得る知識には以下のものが考えられる。

- 競合する知識 : 戦略が複数個ある場合や、代替案がある場合、各々の戦略や代替案は仮説を形成する。
- 仮定的知識 : その知識に対する十分な論拠が得られておらず、真偽の判定が変り得る場合、各々の知識を「信じる」か「信じない」かは仮説となる。
- あいまいな知識 : ある知識の方が他よりも確からしいといった不確実さが存在する場合、各知識とそれらの結合は仮説と見なせる。
- 暗黙知識 : 知識が一部不完全であったり欠落している場合、常識や継承等における default は一種の仮説である。
- 発想的知識 : 未知の状況に出会ったときに既存の知識でそれを説明できない場合、その状況を説明するための知識として仮説を立てる。
- 学習的知識 : 過去の経験を問題解決に利用する場合、類推や帰納により得られた知識は仮説と考えられる。

仮説推論 (hypothetical reasoning) とは、このような仮説となる知識を取り入れてそれに基づいて進める推論であり、いわゆる高次推論機能の多くのものにとって、その実現のための一つの大きな鍵となると思われるため、「基本的推論方式」として出来るだけ自然に実現できる機構が望まれる。

従来の仮説推論の研究は主として、Truth Maintenance System [20] から派生してきたものが多い (e.g. ATMS [10])。 これについては与えられた仮説を基にして推論中における database の一貫性を管理するものであり仮説の与え方までは余り議論されていないし、効率の面での問題は未解決な部分が多い。 またこれとは別に発想推論に基づく

仮設選定〔26, J5〕の研究も行われており、与えられた既知の知識集合が無矛盾であるという前提のもとで、やはり可能な仮説集合を前もって与えておかなければならないが、論理に基づくアプローチとして興味深いものがある。 ところでこれらの研究は相互に関連する部分もあり、全体システムの中では、「仮説の生成・選択・検証」という形で取り入れられるべきであるが、各々がその中の一部分についての技術に留まっている。 多くの問題解決・推論を統合的に仮説推論に基づいて説明できるようにするためには、仮説推論に対して一つの統一的枠組を与えることが必要となるだろう。

1.2 何故仮説推論なのか? (Why)

問題解決における重要な課題の一つに組み合わせ爆発の回避がある。 たとえば、合成問題（設計／計画）、高次推論機能（非単調推論、帰納推論、類推）などは本質的に組み合わせ問題としての性格を持っている。 これに対処するための基礎となる要素は探索（search）である。 一般に問題解決における dynamic な表現を考えてみると、AND/OR グラフとして表わすことができる。 このグラフ上を探索することによる問題解決を考えた場合、例えばルールベースシステムでは強力な heuristics として生成規則が与えられ、OR の部分が決定的に選択されている。 そうではなくて知識が競合しており一意には決定できない場合の OR の処理は、すべての可能な解を捜すことや、ある枝を仮に選んでみて推論を進めもしうまくいかない場合は backtrack して別の解を捜すことが必要となる。 ところが、現在の知識ベースシステムではそのような OR の処理がうまく行われていないのが現状である。 そこで最近注目されている仮説推論を考えてみた場合、まさにこの枠組みにあてはまるため、一つの解決策として AND/OR グラフの探索を仮説推論の枠組みの中で定式化することが考えられる。

現在の expert system shell では仮説推論のための機能が備わっているものは少なく、仮に仮説をユーザが何らかの形で与えたとしても、それを検証するメカニズムが不十分なため、例えば backtrack して代替案を捜すというようなことが簡単には出来ない。現状では、ART の viewpoint 機構〔5〕、KEE の KEEWorld〔39〕に ATMS の機能が一部インプリメントされているが、効率面やその他で実用的と言えるかどうかは疑問である。それ以外ではいずれも研究における実験システムにどどまっている。 このように約10年に亘って研究されてきた幾つかの技術（2章 参照）は残念ながらまだシステムとして実用に堪えるとは言い難い。

仮説推論方式の確立により、知識ベース管理に対する要請が従来と異なってくる可能性もある。 すなわち、矛盾する知識を含んでいても高次推論としての仮説推論によってそれらの知識を扱うメカニズムがカバーされることが期待される。 これにより知識ベース管理における一貫性管理に対する負担を軽減することができ、ひいては仮説推論に合った枠組みで知識を体系化することが要請されるだろう。 従って知識獲得に対する方法論も既存のものとは違ったものとなり得よう。

また、何故仮説推論が必要かという点に関しては何に使われるかという応用面からの検討が重要である。 これについて次に述べる。

1.3 仮説推論は何に使えるか? (Where)

(1) expert system shell or AI の技術としては、

・競合する知識のもとでの推論（互いに矛盾する世界における推論）

・非単調推論 〔38〕
・説明機能の高度化
・knowledge base refinement
・推論過程からの学習(一般化)
等に利用できると思われる。

(2) また応用問題に対しては例として、
・合成型問題 (synthesis) では、
組合せ問題である設計/計画 (generate & test)や、
constraints に基づく問題解決等があり、
・解析型問題 (analysis) では、
故障箇所の検出 (e.g. ATMS を用いた複数故障仮説に基づく診断〔16〕) や、
データ解釈への適用が考えられる。

1.4 如何にして仮説推論にアプローチするか? (How)

現時点では「仮説推論」という言葉自体が新しいものであり、その背景、必要性が統一的に論じられたことも殆ど無いに等しい状況である。 1.1 で述べたように基本的推論方式として仮説推論が定式化され、統一的な枠組が与えられれば、それに越したことはない。 しかし、多くの未解決な問題 (3, 4 章 参照) を全て説明できるようにするには多大の努力を必要とする。 従って、全体システムの構成を考えつつ、扱える仮説の種類や推論の機能を絞って適切なアプローチをする必要がある。

また、実用化へのアプローチとしては、仮説推論が対象とする応用問題の中で何に対して最も有効かを考え、その点に関して取り入れていく方法も必要となるだろう。 例えば、単に現在提案されている TMS や ATMS をシステムに組み込んだだけでは組合せ爆発の問題が起きる。 それは ART の hypothesize コマンドの使用法すら十分理解されているとは言い難いことからも指摘することができる。 この理由として ART の機能が十分実用レベルに達していないこともあるだろうが、設計方針として現実の問題への応用が熟考されていたのだろうかという反省もする必要がある。

上で挙げたことに加えて、仮説推論に対するアプローチとして以下の問題を考慮することは重要である。 現在これらの問題に対する方針を考案中であるがその一部を 5 章で述べる。

(1) 仮説推論をどのレベルでサポートするべきか?

例えば、ART の hypothesize のようなコマンドとしてサポートするのか? それとも基本的な推論メカニズムとしてサポートするのか? 但し、前者の場合でも単に、従来のシェルの拡張機能では済まず dependency の記録等でアーキテクチュアから考え直さなければならない。

この問題は「何を仮説と見なすか?」という問題と密接に関わっている。

仮説には、
・ユーザが与える仮説
(何が仮説であるかを明示してシステムに与える。)
・システムから見て意味のある仮説
(ユーザは仮説と意識していないがシステムが機能、効率面で仮説として扱

って処理するもの：例えば、 $x \in \{1, 2\}$ において、$\{x = 1\}$, $\{x = 2\}$ 等の仮説を生成する。）

の二通りが考えられる。 前者であれば、assertional language やコマンドとしてサポートすれば十分であるが、後者の場合だと非決定性を有する選言的知識（OR）はすべて仮説と考えて処理することが考えられるため根本的に推論メカニズムの構成から考えていかなければならない。

さらに、この問題は「仮説の生成・選択・検証をすべてシステムで行うかどうか?」という問題とも関係する。 特に仮説生成は何を仮説にするかによって大きくアプローチの仕方が変わる。

(2) (1)とも関連するが、知識表現形式との連携をどのように行うか?

できるだけ仮説を自然な形で表現するにはどうすればよいか? 例えば、メタレベルで記述できることも一つの方法である。 またそれが仮説であると気にしないでも記述できるようにするにはどうすればよいか? さらに、time や action の記述はどうすればよいのか?

(3) 仮説に基づく問題解決システムにおいて、problem solver（推論部）と TMS を明確に分離するべきかどうか? また分離するとしたらどのように行えばよいのか?

(4) expert system shell の中に仮説推論を組み込むとした場合、説明機能や知識ベース管理との連携をどうすればよいか?

(5) 大規模・複雑な問題を扱う場合、階層的問題解決との関連はどうなるのか? このような場合、問題解決において仮説は様々なレベルで考えなければならないが、それらを同一の仮説組合せで考えることは自然ではない。 例えば、ART の場合は二重レベルの viewpoint が扱えるが、これはやや特殊であるため一般には一重レベルで解いている。 これは階層的に問題を分割して解きたい場合には不都合であることが多い。 また、あるレベルでは仮説推論を取り入れたいが、そうでないレベルも存在するかもしれない。 一体、階層的な仮説推論とはどのような形にするのが望ましいのかを考えることは実用的な観点からは意義が大きい。

(6) 仮説推論を応用問題に適用する場合、次のことを明らかにする必要がある。

・何が仮説となり得るか?

・仮説間の優先順位はどのように表現し得るか?

・何をもって仮説を真あるいは偽であると判定するか?

・求める解は全解が必要か、(準)最適解が必要か、単に一つ解が求まれば良いのか?

・何が矛盾か（矛盾をどのように表現し与えるか）? 矛盾が生じたときに、それをデータベースにどのような形でストアしておくか?

・矛盾の記録を後の推論にどのように活用するか?

・TMS にはデータ間の dependency を記録するシステムが多いが、これには非常に多くの記憶領域が必要となるため、実際には必要となる部分のみ記録を取る等の方法が考えられるが、どこで必要となるのか?

2. Review of Existing Work

仮説推論に関連する従来からの研究の例を以下に挙げる。 これらの研究内容の大まかな推移を Fig.1 に示す。 また、各々の研究内容の相関関係の概念図を Fig.2 に示す。各々の詳細については BIBLIOGRAPHY に示した各文献を参照されたい。

(1) Truth Maintenance System
- EL [49]
- AMORD [13, 14]
- Doyle's TMS (RMS) [19, 20, 22]
- McAllestar [36]
- WATSON [29]
- Multiple Belief-space Reasoner (MBR) [33, 34]
- McDermott [4, 37]

(2) Assumption-based TMS
- de Kleer's ATMS (CMS) [8, 10, 11, 12, 15, 17]
- SLP [7]
- ART [5, 51]
- KEE [39, 40]

(3) Intelligent Backtracking
- Prolog [2, 35]
- Constraint satisfaction problem [6]

(4) Belief System
- Theories of reasoned assumptions [21]
- Belief revision [47, 52, J3]
- Model of belief [32]
- Knowledge equilibration [J2]

(5) Multi-valued Logics
- Conterfactuals [27, J7]
- Ginsberg [28]
- Boolean-valued Prolog [53, J6]

(6) Non-monotonic Reasoning [24, 25, 38, 45]

(7) Abduction
- RESIDUE [26]
- Theorist [42, 43, J1]
- Hypothesis selection system [J4, J5]

(8) Learning
- Learning by analogy [3]
- Soar [30]

(9) Knowledge Base Refinement [48]

(10) Applications
- Qualitative reasoning [9, 50]
- Diagnosis [1, 16, 23, 41, 46, J8]
- Planning [18, 31, 44]

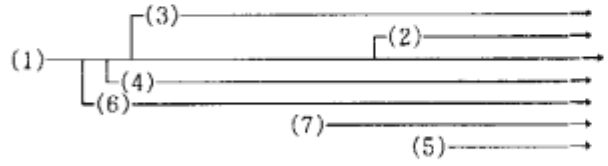

Fig. 1 Historical view of hypothetical reasoning

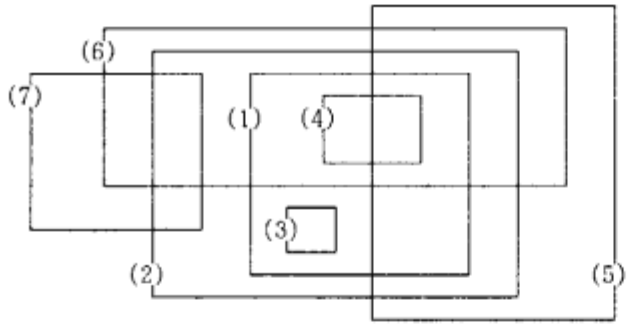

Fig. 2 Classification of hypothetical reasoning

3. Basic Concepts for Hypothetical Reasoning

仮説推論に必要となる技術を問題点も含めて以下に挙げる。

(1) 仮説生成

・仮説空間（意識的 / 無意識的 に与えられている）から、意味のある要素をもつ仮説集合を生成する問題である。
・仮説の表現方式が問題となる。
・TMS の場合、仮説は既に与えらていることが多い。 すなわちユーザが何らかの形で与える。 そのときどのように与えるべきか。
・デフォルト推論においては default 知識が仮説となる。 この場合も、そのような知識を表現する枠組を与える必要がある。
・RESIDUE や Theorist においては仮説空間から発想的推論を基に仮説を生成する。これも仮説空間が限定されている必要がある。 また、これらの仮説生成のレベルは実は(2)の仮説選択と見なすことができる。
・問題によっては、領域固有の知識や深い知識をうまく利用することで、可能な仮説集合を自動的 に推論することも考えられる。 例えば、診断では構造と機能の知識から仮説集合を生成し、 設計では部品の組合せから仮説集合を生成する。

(2) 仮説選択

・仮説集合からどの仮説を候補として選択するかという問題である。
・(1)の仮説生成と合わせて、集合の選択問題として捉えることができる。
・(3)の仮説検証と合わせて、仮説の展開の順序による control の問題でもある。
・ explicit に仮説間の preference（半順序）を与えることにより効率的に制御することも考えられる。
・いずれにせよ、at rundom や exhaustive な選定を行っていたのでは、 効率が悪い。組合せ爆発を如何に最小限に抑えるかが問題となる。

(3) 仮説検証

・選択された候補仮説を展開して真偽を判定する問題である。
・一つの context に固執する場合（TMS）、すべての contexts を同時に展開する場合（ATMS）等がある。 これらの何れが良いかという議論は扱う問題の性質によって議論されるべきものであり、一概には言えない。
・何をもって仮説を真あるいは偽であると判定するか。
・矛盾をどのように表現し与えるか。 矛盾が生じたときに、それを database にどのような形でストアしておくか。 またそれらの記録を後の推論にどのように活用するか。
・この問題も膨大な探索空間を如何に効率良く探索するかが問題である。コストのかかる計算は後回しにして、多くの plausible path を展開することも考えられる。

(4) データ依存関係の記録

・データや規則間に dependency の記録を持たせているシステムが殆どである。
・矛盾の検出や backtracking の際に有効に使う。
・新事実の追加が database に及ぼす影響は非単調である。 この場合 justification は default と同様の機能を果たす。 ある信念が in や out になると他のデータがそれにつられて信念の状態を変化させられる。 (belief revision)

・TMS による非単調論理は、多くの場合直観的であり、厳密な記述が少ない。
・非常に多くの記憶領域を必要とする。 そのため、必要となる部分のみ記録を取る等の方法が考えられないか。

(5) 知的 backtracking
・失敗に直接関係のある時点（仮説選択時）に戻る。
・同様のパターンによる失敗は繰返し起こさないようにする。
・ATMS では多重 contexts を持ち、かつ相矛盾していてもよい contexts が同時に展開されるため、backtracking は存在しない。 しかし、矛盾に関する知識を反映させることにより、以降の推論の効率化に役立てなければならないことに変りはない。
・(2), (3)と同様に control の問題を考えなければならない。

(6) Implementation techniques
・TMS の context switching 〔37〕
・AMORD の explicit control 〔14〕
・MBR の wff による表現と control 〔35〕
・Prolog の intelligent backtracking 及び AND-parallelism 〔2〕
・ATMS の environment lattice 〔10, 11, 17〕
・ATMS-problem solver における consumer 〔12, 17〕
・ART, KEE の viewpoint (World) 機構、及び merge 機構 〔39, 51〕
・ART の temporal representation (shadowing) 〔5〕

4. Problems for Hypothetical Reasoning

3 章で述べた技術の中から特に問題となっており、かつ現在個人的に興味のある部分と、それに対する私の考え方は以下の通りである。

(1) TMS の効率化 (control)

Doyle の TMS [20] は仮説を含むデータの状態を in (信じられている)・out (信じられていない) の二通りで表わして管理し、矛盾が生じたときには dependency に基づいて backtrack すること (dependency-directed backtracking) により database の一貫性を保持する機構であるが、backtracking 及び仮説の選択における制御 (control) を知的に行う方法は与えていない。また、de Kleer の ATMS [10] は TMS を仮説の組合せに基づいて多重 contexts に拡張し backtracking を回避しているが、組合せ爆発の問題に対しては解決策を与えるには至っていない。従って、これら TMS の上に知的な制御機構を実現することが大きな課題として挙がっている。

〔アプローチ〕

① 探索 (search) の適用

・最良優先探索 (best-first search)

仮説間に何らかの評価が与えられるときは、TMS の control の問題は search の手法を適用することで対処することが考えられる。また、特別な評価関数が与えられていなくても、特に ATMS のような多重 contexts を扱う場合を考えると、仮説が競合しているときにはそれらが OR の関係にあると見なせば、AND/OR グラフ上の探索問題として考えることができる。すなわち、TMS / ATMS を探索機構を備えた推論システムとして議論できる。一般に仮説推論のような組合せ問題に対して、縦型探索 (depth-first search) や横型探索 (breadth-first search) のような方法で全解探索を行っていたのでは何の解決にもならない。人間の問題解決過程との類比で言えば、何らかの heuristics を基にして plausible path を展開していくような best-first search を取り入れることが必要とされよう。

・評価関数を用いた枝刈り

最初に、人工知能研究の基礎的な成果である B&B (branch and bound) や α-β との関連に着目してみる。一般に TMS では制約条件 (constraints) を矛盾の検出に使用しているが、これを不要な仮説の枝刈り (pruning) のための限定要因 (bound) として使用することが考えられる。すなわち、constraints の dynamic な変化で不要な仮説の枝刈りを行う。

> 例：job shop scheduling における工程時間は得られた計画の満たすべき基準として矛盾の検出に使用できるが、これを得られている最短工程時間によって、その時間内で終了しないことが判った計画を枝刈りするために使用することにすれば、一層の探索の効率化を計ることが可能である。

ここで best-first serach と結び付けて考えてみる。ATMS では multiple world とはいっても sequential に実行するのだから、最も好ましい仮説世界 (plausible world) から順に選択していくことにより、解に早く到達することができ、効率的に探索を行わせることが期待できる。このとき、backtracking が効率を上げる。ところで、ATMS の control については de Kleer も一般的なことは余り言えないとしながらも、node の label において most general environment が早く見つかるほど余分

な label の updating を除去できることと、contradiction の most general な形態が 早く見つかるほど究極的に inconsistent である environments に関係する問題解決のステップを削減できるために効率がよいとしている。 この考え方でいくと探索法は良さそうな path を展開していく best-first search というよりは、早く矛盾が見つかるようにする "worst-fist search?" が良いことになる。 この問題を考えてみることは非常に興味深い。

DDB (dependency-directed backtracking) では contradiction が起こったときにのみ backtrack する。 これに対し、best-first search での backtracking は他に良さそうな候補解がある場合の context switcing に相当する。 従って、backtrack して戻る前の path を取っておく必要がある。 また、評価関数の値が更新されること (updating) があるとすればそれは path の切換えを誘引する。

まとめると、backtracking に対する考え方は次のようになる。

backtracking
- contradiction → fail (矛盾) …… defeasible
  → optimal でない… non-optimal
- context switching → 現在の状況により他の path を優先

・ヒューリスティック探索

評価関数が、解が consistent であるための十分条件を有するとき、その条件を満足する評価値を heuristics として用いることが考えられる。 このことにより、解に到達しやすいと思われる仮説から選定が行われることになる。 これに対し、basic な ATMS では矛盾が検出されるまでは何ら戦略がない。

・explicit control

de Kleer は DDB の ATMS への組み込みにより、必ずしも全解が必要とされない問題に対して効率良く仮説空間を探索するための単純な手法 [17] を提案している。
この方法では ATMS が扱う multiple world に対する environment lattice の方式に、DDB によって線型に探索することを取り入れている。 しかし、複数の並行した仮説世界上では、単なる depth-first search よりも、上で述べた B&B による best-first search を組み入れたほうがより ATMS の特徴が発揮され、さらに効率化が期待されることが考えられる。 また、仮説選択順序のより良い指定が効率を左右するが、評価関数が求まるときには、上記の方法が適用できるし、さもなければ下記の方法が適用できる。

・評価関数として適切なものが得られない場合

評価関数を使う代りに 'dominance' を使う。

ADB : A の方が B よりも良い (と思われる)。 (→ preference)

このうち最も簡単な場合は全仮説間に全順序が与えられている場合であり、仮説選択の順序として使える。 また、難しいのは部分的に半順序関係が与えられる場合であるが、これには multi-valued logics の適用 ((3) を参照のこと) も考えられよう。

② 記憶空間の効率化

・部分的な dependency の記録

一般に dependency の記録は非常に多くの記憶領域が必要とされるため、ユーザが指定した部分のみ justifications を取ることができることが望ましい。 例えば階層的に問題を分割して解く場合では、ある部分問題は仮説推論が必要であるが、別の部分問題では必要ないかもしれない。 このようなときは部分問題毎に指定できることが

望ましい。

これをさらに発展させると、問題の中で cost のかかりそうな部分をシステムが検出し、その部分のみ dependency を取るような自動制御も考えられる。

・同一状態の縮退 (merging)

記憶領域を有効に利用するために必要となる技術の一つに merging がある。これは ATMS では複数の environment による label づけとして表わせるが、ART [5] や KEE [39] ではさらに auto merge によって problem solver の意図を超越していると思われる程自動的に merging を行う機構が備わっている。これは、特に時系列変化や状態変化を扱う時に有効と思われる。例えば計画における planning-sequence の処理に向いている。

(2) CSP (constraint satisfaction probrem) への応用

設計や計画のような合成型の問題を扱う場合、制約 (constraints) の扱いが鍵となる。この問題に対して CSP として研究がなされているが、近年は control strategy として backtracking を適用することが検討されている。すなわち dead-end に陥った原因を解析し、記憶しておくことによって、後の決定の guide を行い、同様の失敗を引き起こさないように有効に使用する。ここに TMS のアプローチが適用できる [6]。

また一般に制約を扱う場合に困難とされているのは、動的な constraints の扱いである。これには次の二通りが考えられる。

・weak constraints の relaxiation

・途中で作り出される constraints

これに対処する一つの方法としては、その constraints が active か passive かで仮説世界を形成することが考えられる。この問題に対しても (1) と同様に control が重要である。

(3) 論理に基づく定式化

・発想推論 (abductive reasoning) による説明

述語論理と導出法によって仮説推論を捉えようとするとどうなるだろうか。その一つの試みとして発想推論があり、RESIDUE [26] や Theorist [43] では与えられた仮説空間から候補仮説を適当に選び、目標の節集合との refutation によって仮説の正当性を検証しようとしている。これらには、失敗の記録を有効に利用するという思想はなく、TMS とは別のアプローチと考えられるが、仮説空間を予め用意する必要があるため扱う問題が単純過ぎることから、実用的なシステムとは言い難い。但し、仮説推論を非単調論理として定式化するには向いているため、研究する価値は大きい。また de Kleer も ATMS を一般化した CMS [15] において発想推論への適用の可能性を論じている。

・多値論理 (multi-valued logics) に基づく定式化

logic に基づくアプローチの中でも特に注目されるのが多値論理に基づくものである。Ginsberg は反事実条件文 (counterfactuals) [27] と ATMS との類似点を指摘しており、またそれを発展させた多値論理 [28] において、新しい事実が加わったときに TMS の機能を実現する方法を述べている。これと類似の研究に Boolean-valued Prolog [53] があり、仮説推論への応用の試みも一部行われている。

これは例えば次のような場合に有効に使えると思われる。unknown, ambiguous な知識に対して t (true) と f (false) の間の任意の値を与えると、それは lattice 構造を形成する。これにより alternatives の間の preference を半順序関係として記述することが考えられる。

・Logic Programming とその関連

Prolog の素朴な backtracking を効率化しようとする intelligent backtracking の研究も一部で行われている〔2, 35〕。これは TMS を top-down 的に適用しようとするものだが、実用的なプログラミング言語にまでは至っていない。また、次の図式を考えてみるのも面白い。

Problem Solving		Logic (Prolog)
Doyle's TMS	←→	intelligent backtracing
ATMS	←→	ここに来るものは何か?

一方で、parallelism に対して、TMS はどのように考えればよいのだろうか。例えば AND-parallel による Horn clause の実行は intelligent backtracking で実行される順序を ignore して fail の履歴を使うことへと展開可能であると考えられる。また、ATMS への parallelism の適用は global database の処理が問題となる。

(4) learning への展開

あいまいな知識を用いて問題解決を行う場合、最初は search のような weak method を用いて解を発見することを試みるが、TMS で矛盾の記録を行うと以後の戦略にその知識を積極的に利用して探索空間を縮小していくことになる。これをさらに発展させると、特定の値で fail したときに、その値あるいは矛盾を引き起こした原因を可能な限り一般化しておき（失敗の一般化）、以降の推論を効率化する「失敗からの学習機能」となる。ここではそれを generalizing while backtracking と呼ぶことにする。

generalizing while backtracking とは仮説が失敗した究極の原因を追及し、backtrack するときに一般化するものである。この機能により、その問題解決の以降では、仮説の絞り込み・早期枝刈りに利用（search の効率化）でき、次回以降の実行に活用するならば、いわゆる chunking メカニズムとなる。

これまでに行われている関連研究としては、CSP の探索における learning〔6〕がある。他に仮説推論とは直接の関係はないが justification の記録を利用した learning by analogy〔3〕や chunking in Soar〔30〕等がある。

5． An Overview of Hypothetical Reasoning System

4 章での問題点を踏まえた上で、現在検討を行っている「仮説推論システム」（仮称）(hypothetical reasoning system) の構想を述べる。

〔 仮説 〕

本システムでは「仮説」の概念を次のように定義する。 厳密には「真」を true と in (= default true) とに、また「偽」を false と out (= default false) とに別けて考えなければならないが、ここでは簡単のため次のように説明しておくに止める。

・正当化 (justification) ： 知識 a が知識 b によって導かれるとき、
「b は a によって正当化される (justified)」という。

・仮説 ： 次の二つの性質を満足する知識を「仮説」という。
(1) その知識自身によって正当化されている。
(2) 恒真である（すなわち必ず成立する）知識、及びそれらの知識から導かれる知識だけからは正当化されていない。

本定義によれば、(1) によってその知識が仮説として明示的に導入されていることを示し、(2) によってその知識が「真」である事が保証されていないことを示す。 (1) の性質により、ある仮説から導かれていてそれが自身によって正当化されていないものは仮説とは見なさない。

〔 システム構成 〕

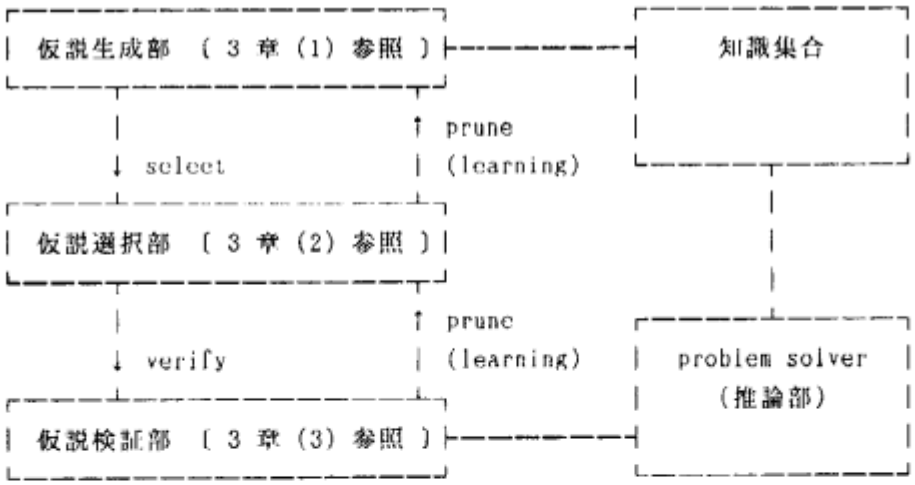

Fig. 3 An architecture of hypothetical reasoning system

・仮説生成部 ： 知識の集合から仮説空間を決定し、その中から問題解決において意味のある要素から構成される仮説集合を生成する。
領域知識、深い知識を利用して生成する。
生成は一括、あるいは段階的、階層的に行われる。
場合によってはこの部分は省略されてユーザまたは problem solver

が陽に与えることもある。

・仮説選択部：仮説生成部から与えられる仮説集合から、仮説となる要素を候補として選択する。 また仮説検証部で得られた結果がフィードバックされて、不要な仮説を選択することを回避する。

・仮説検証部：仮説選択部で選択された仮説に基づいて推論を展開し、その正当性を判別する。 その際、TMS によって database の多重 contexts の各々に対して矛盾が生じないように dependency に基づいた管理を行い、仮説選択部に対して control のための指示を与える。

推論の展開に際して、problem solver が weak method (search) を適用することができ、効率的に control できる。

〔 仮説推論に対する研究方針 〕

現在、統一的な枠組みを持つ仮説推論システムの構想を立てているが、その実現に向けては多くの問題点を解決していかなければならない。 その第一歩として仮説推論に基づく問題解決システムの知的な control の問題を取り上げ研究を進めている。

今後は次のステップを踏んで徐々に拡張していきたい。 但し、(1) の次にはどれから始めても構わないが、現在の興味は主として (2) にあり、これについては 6 章で現在考えていることを一部述べる。

(1) 最初はユーザが仮説を陽に与える。 何等かのコマンドまたは assertional language によって仮説集合または仮説要素を表現して problem solver を通して与える。

但し、仮説の表現形式や、justification の記録を含めて従来のシェルとはアーキテクチュアから違ったものとなる。 また、(2) への展開のために多重 contexts を考慮した ATMS 的なアプローチを取る。

(2) 次に選言的で非決定的な知識に対してシステムが自動的に仮説として扱う枠組みを構成する。 これにより、AND/OR 木 上の問題解決は仮説推論の枠組みの中で扱われるため、仮説推論が基本的推論方式として位置付けられる。 ここでは特に search と関連づけて捉えることを試みる。 また、このステップで考えるアルゴリズムに対して、非単調推論や logic の観点から考察を行う。

(3) 与えられた知識集合から領域固有の知識や戦略を利用することによって、自動的に仮説生成を行う仮説生成部を実験的に作ってみて、それをできるだけ汎用化したものへと洗練する。 これには generic task のようなアプローチが有効であると思われる。すなわち、仮説生成部は扱う問題のタイプに依存したものとなる。

(4) 問題解決における failure や success の過程を次回以降の推論に役立てるように一般化する機能を考える。 仮説推論と学習の親和性について議論をしたい。

(5) シェルとして十分実用に堪えるための洗練、改良を行う。 その際、説明機能の高度化など仮説推論の要素技術から派生される機能も盛り込んだものとする。

〔 応用問題からの検討 〕

当面、設計/計画問題を考慮する。 generate & test は仮説の生成・選択と検証として仮説推論そのものの枠組みの中で捉えることができる。 また、constraints の扱い、再設計 (failure recovery)、試行錯誤過程等において、それらと仮説推論との相関を明らかにしたい。

6． An Approach toward intelligent control strategy for Hypothetical Reasoning

4 章 (1) 及び 5 章 に関連して現在考えている事項の詳細を一部述べる。 de Kleer の ATMS の多重 contexts を参考にした hypothetical reasoning system において、効率良く control 出来るように ATMS を再構築するための方法について論じる。

de Kleer の ATMS における environment lattice [10] は所謂 Hasse diagram を上手く利用して探索を行っているが、この手法は ATMS の最大の特質を multiple contexts の展開にあるとみるならば、 ATMS の本質というよりは一つのインプリメントの技法と考えるべきである。

ところが、ATMS には 4 章 (1) でも述べたように control に対しては議論する余地が多く残されている [12, 17]。 従って現在は、多重世界 (multiple world) の考え方に対して、効率的な手法を独自に考えており、特に search と絡めた議論を展開している。

multiple world は基本的に AND/OR 木 (あるいは AND/OR グラフ) 表現が可能である。従って、AND/OR 木探索のアルゴリズム (AO*, SSS*, GBF, GBB 等) が適用できるはずである。 このとき、一つの solution tree (OR の枝のうちの 1 つ、AND の枝は全てを持つ subtree) が 一つの仮説世界に相当する。

そこで、仮説世界の展開を AND/OR 木 探索に見たてて解く。 4 章 で述べたように評価関数が使えれば B&B を併用する。 ところで、GBB によると B&B で AND/OR 木探索も表現できる。 従って、仮説世界 = AND/OR 木、問題解決 = B&B、により汎用的に仮説推論が表現できると考えられる。

ポイント

1. 評価関数が使える時は、そのまま search に反映できるため、
   Assumption-based DDB (DDB + environment lattice)
   以上の効率化が可能である。
2. それ以外のときは、environment lattice 以上の性能を引出す必要がある。
   このとき、AND/OR 木 探索も space を非常に食うため、resource との関連で考慮する必要がある。
3. inclusive OR に対する特別な配慮(探索アルゴリズム)が必要である。
   通常、AND/OR 木における OR は oneof disjunction すなわち、exclusive OR であるため、その仮説が成立するかどうかという選択とは異なる。
4. 矛盾に導く知識を積極的に利用して top-down 的に問題解決を行う。
   そのとき矛盾の情報をどのように持たせるか。
5. ある種の導出法が必要となる。 例えば、ATMS では justification として Horn clause 以外に複数の positive clauses を扱えるように負超導出 (negative hyper-resolution) を使用している [11, 17]。 但し、探索においては、
   「解が存在するためには、仮説は～の条件を満たさないといけない。」,
   「 n 個の内 (n-1) 個は矛盾を導くとすれば、残りの 一個が解に違いない。」
   のような、一種の「消去法」として導入すれば、pruning に効果がある。 但しこの場合、前提として「必ず解が存在する。」という仮説が導入されている。

7. Conclusion

仮説推論に対する考え方とその必要性、現在の研究状況と問題点、さらに現在考えている仮説推論システムの概要について報告した。 高次推論機能の実現のためには仮説推論が一つの大きな鍵となると思われるため、基本的推論方式として研究の整備が望まれている。 また、応用面から考えてみた場合、設計や計画といった合成型の問題や、定性推論を用いるようなさらに高度な解析型問題を対象とするエキスパートシステムの実現のためにも仮説推論は必要不可欠な技術である。

このように仮説推論が重要であるにも拘らず、約 10 年に亘って研究されてきた幾つかの技術は残念ながらまだ実用に堪えるとは言い難いものである。 従って、仮説推論に対して過度の期待を持つこと無く適切なアプローチを行うことが必要である。 例えばある応用領域を考え、その中に仮説推論の技術を何等かの形で導入していくことにより、実用的な技術が生み出されることは十分考えられる。

現在、統一的な枠組みを持つ仮説推論システムの構想を立てているが、その実現に向けては多くの問題点を解決していかなければならず、その第一歩として仮説推論に基づく問題解決システムの効率的な制御の問題を取り上げ研究を進めている。 そして、応用としては当面合成型の問題について考えて行く予定である。

BIBLIOGRAPHY

1. Brown, J.S., Burton, D. and de Kleer, J., Pedagogical, Natural Language and Knowledge Enginnering Techniques in SOPHIE I, II and III., in: Intelligent Tutoring Systems, edited by Sleeman, D. and Brown, J.S., (Academic Press, 1982) 227-282.
2. Bruynooghe, M. and Pereira, L.M., Deduction Revision by Intelligent Backtracking, in: Current Issues in Prolog Implementation, edited by Campbell, J.A., (Wiley, 1984) 194-215.
3. Carbonell, J.G., Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition, CMU Computer Science Dep., Technical Report CMU-CS-85-115, 1985.
4. Charniak, E., Christopher, K.R. and McDermott, D., Artificial Intelligence Programming, Lawrence Erblaum, 1980.
5. Clayton, B.D., ART Programming Tutorial, Inference Corporation, 1985.
6. Dechter, R. and Pearl, J., Learning while Searching in Constraint-Satisfaction-Problems, AAAI-86 (1986) 178-183.
7. d'Ambrosio, B., Truth Maintainance with Numeric Certainty Estimates, Proc. 3rd Conf. on Artificial Intelligence Applications (1987) 244-249.
8. de Kleer, J., Choices without Backtracking, AAAI-84 (1984) 79-85.
9. de Kleer, J., How Circuits Work, Artificial Intelligence 24 (1984) 205-280.
10. de Kleer, J., An Assumption-based TMS, Artificial Intelligence 28 (1986) 127-162.
11. de Kleer, J., Extending the ATMS, Artificial Intelligence 28 (1986) 163-196.
12. de Kleer, J., Problem Solving with the ATMS, Artificial Intelligence 28 (1986) 197-224.
13. de Kleer, J., Doyle, J., Ritch, C., Steel, G.L. and Sussman, G.J., AMORD: A Deductive Procedure System, MIT AI Lab., Memo No. 435, Jaunary 1978.
14. de Kleer, J., Doyle, J., Steel, G.L. and Sussman, G.J., Explicit Control of Reasoning, MIT AI Lab., Memo No. 427, June 1977.
15. de Kleer, J. and Reiter, R., Foundation of Assumption-based Truth Maintainance Systems: Preliminary Report, to appear in AAAI-87, 1987.
16. de Kleer, J. and Williams, B.C., Diagnosing Multiple Faults, Artificial Intelligence, forthcoming, 1986.
17. de Kleer, J. and Williams, B.C., Back to Backtracking: Controlling the ATMS, AAAI-86 (1986) 910-917.
18. Dhar, V., An Approach to Dependency Directed Backtracking using Domain Specific Knowledge, IJCAI-85 (1985) 188-190.
19. Doyle, J., Truth Maintenance Systems for Problem Solving, MIT AI Lab., AI-TR-419, January 1978.
20. Doyle, J., A Truth Maintenance System, Artificial Intelligence 12 (1979) 231-272.

21. Doyle, J.,
Some Theories of Reasoned Assumptions: An Essay in Rational Psychology,
CMU Computer Science Dep., Technical Report CMU-CS-83-125, 1983.
22. Doyle, J., The Ins and Outs of Reason Maintenance,
IJCAI-83 (1983) 349-351.
23. Doyle, R.J., Constructing and Refining Causal Explanations from an
Inconsistent Domain Theory, AAAI-86 (1986) 538-544.
24. Etherington, D.W., Formalizing Nonmonotonic Reasoning Systems,
Artificial Intelligence 31 (1987) 41-85.
25. Feldman, Y.A. and Rich, C., Reasoning with Simplifying Assumptions:
A Methodology and Example, AAAI-86 (1986) 2-7.
26. Finger, J.J. and Genesereth, M.R., RESIDUE: A Deductive Approach to
Design Synthesis, Stanford HPP, Memo HPP-85-1, 1985.
27. Ginsberg, M.L., Counterfactuals,
Artificial Intelligence 30 (1986) 35-79.
28. Ginsberg, M.L., Multi-valued Logics, AAAI-86 (1986) 243-247.
29. Goodwin, J.W., WATSON: A Dependency Directed Inference System,
Proc. AAAI Workshop on Non-monotonic Reasoning (1984) 103-114.
30. Laird, J.E., Rosenbloom, P.S. and Newell, A.,
Chunking in Soar: The Anatomy of a General Learning Mechanism,
Machine Learning, Vol.1 No.1, January 1986.
31. Latombe, J.C., Failure Processing in a System for Designing Complex
Assemblies, IJCAI-79 (1979) 508-515.
32. Levesque, H.J., A Logic of Implicit and Explicit Belief,
AAAI-84 (1984) 198-202.
33. Martins, J.P. and Shapiro, S.C., Reasoning in Multiple Belief Spaces,
IJCAI-83 (1983) 370-373.
34. Martins, J.P. and Shapiro, S.C., A Model for Belief Revision,
Proc. AAAI Workshop on Non-monotonic Reasoning (1984) 241-294.
35. Matwin, S. and Pietrzykowski, T.,
Intelligent Backtracking in Plan-Based Deduction,
IEEE Trans. Pattern Anal. Machine Intelligence 7, No.6 (1985) 682-692.
36. McAllester, D., An Outlook on Truth Maintenance,
MIT AI Lab., Memo No. 551, 1980.
37. McDermott, D., Contexts and Data Dependencies: A Synthesis,
IEEE Trans. Pattern Anal. Machine Intelligence 5, No.3 (1983) 237-246.
38. McDermott, D. and Doyle, J., Non-monotonic Logic I,
Artificial Intelligence 13 (1980) 41-72.
39. Morris, P.H. and Nado, R.A., Representating Actions with an
Assumption-Based Truth Maintenance System, AAAI-86 (1986) 13-17.
40. Nardi, B.A. and Paulson, E.A., Multiple Worlds with Truth Maintenance
in AI Applications, Proc. ECAI-86 (1986) 437-444.
41. Peng, Y. and Reggia, J.A., Plausibility of Diagnostic Hypotheses:
The Nature of Simplicity, AAAI-86 (1986) 140-145.
42. Poole, D., A Logical System for Default Reasoning,
Proc. AAAI Workshop on Non-monotonic Reasoning (1984) 373-384.

43. Poole, D., Aleliunas, R. and Goebel, R., Theorist: A Logical Reasoning System for Defaults and Diagnosis, in: Knowledge Representation, edited by Cercone, N.J. and McCalla, G., (Springer-Verlag, 1985).
44. Rao Padala, A.M., Biswa, G. and Bose, P.K., Assumption Based Reasoning Applied to Personal Flight Planning, Proc. 3rd Conf. on Artificial Intelligence Applications (1987) 266-271.
45. Reiter, R., A Logic for Default Reasoning, Artificial Intelligence 13 (1980) 81-132.
46. Reggia, J.A., Nau, D.S., An Abuductive Non-Monotonic Logic, Proc. AAAI Workshop on Non-monotonic Reasoning (1984) 385-395.
47. Rose, D. and Langley, P., Chemical Discovery as Belief Revision, Machine Learning Vol.1 No.4 (1986) 423-451.
48. Smith, R.G., Winston, H.A., Mitchell, T.M. and Buchanan, B.G., Representation and Use of Explicit Justifications for Knowledge Base Refinement, IJCAI-85 (1985) 673-680.
49. Stallman, R.M. and Sussman, G.J., Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis, Artificial Intelligence 9 (1977) 135-196.
50. Williams, B.C., Doing Time: Putting Qualitative Reasoning on Firmer Ground, AAAI-86 (1986) 105-112.
51. Williams, C., Managing Search in a Knowledge-based System, Inference Corporation, unpublished, 1985.
52. Winslett, M., Is Belief Revision Harder Than You Thought?, AAAI-86 (1986) 421-427.
53. Yamaguchi, J., Boolean-valued Prolog —the philosophical background—, ICOT FAI WG, unpublished, 1986.

In Japanese :

J1. 赤間 清, 仮説探索システム HYPOSE, 情報処理学会知識工学と人工知能研究会資料 49-2 (1986) 1-8.
J2. 北上 始, 国藤 進, 古川 康一, 宮地 泰造, Prolog による Belief System の実現方法, ICOT Technical Memo TM-052 (1984).
J3. 久野禎子, 信念の修正機構を応用した物語理解システム, 人工知能学会誌, Vol.2 No.1, 1987年3月.
J4. 国藤 進, 仮説推論, 人工知能学会誌, Vol.2 No.1, 1987年3月.
J5. 国藤 進, 鶴巻 宏治, 古川 康一, 仮説選定機構の一実現法, 人工知能学会誌, Vol.1 No.2, 1986年12月.
J6. 森下 真一, 沼尾 雅之, 広瀬 紳一, 自由ブール代数値をもつ論理型言語の基礎, 日本ソフトウェア科学会第3回大会論文集 D-5-3 (1986) 277-280.
J7. 横溝 博文, 溝口 文雄, 構造情報を用いた論理回路診断システムの試作, Proc. the Logic Programming Conference '86 5.2 (1986) 51-58.
J8. 和田 慎一, 古関 義幸, 故障診断における仮定に基づいた推論方式, Proc. the Logic Programming Conference '86 5.3 (1986) 59-65.