ICOT Technical Report: TR-314

TR-314

仮説推論による探索木の枝刈りについて

井上克巳

October, 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

仮説推論による探索木の枝刈りについて

Pruning Search Trees with an Assumption-based Reasoning

井上 克巳

(財) 新世代コンピュータ技術開発機構

問題解決において複数のコンテキストに沿って推論を進める場合の一般的なアルゴリズムについて考察する。コンテキストは仮説推論における仮説の組み合わせとして表現され、一方、ＡＮＤ／ＯＲ木探索においては一つの解木(solution tree)を構成する。矛盾を導くコンテキストは Truth Maintenance 機構を用いてチェックすることができ、これにより探索木の枝刈りを効率的に行うことができる。

１．はじめに

すべてのＡＩシステムにおいて探索は必要不可欠である。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おり、組み合わせ爆発の回避は重要な課題である。一般に問題解決の過程はインクリメンタルに AND/OR 木を構築しながら同時に探索しているとしてモデル化することができる。例えばこの木上の探索として、従来のルールベースシステムを捉えると 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、非決定処理が必要とされバックトラッキングの技術が用いられる。いわゆる chronological backtracking の冗長な計算と同じ失敗の繰返しを避けるために、知的キャッシュを導入したのが dependency-directed backtracking (DDB) [7] であり、DDB に基づく探索を dependency-directed search (DDS) といいＡＩシステムにおいて近年は一般的な推論制御手法となっている。本稿では、DDS のメカニズムを論理的に解釈することにより、従来提案されていた方式の問題点を指摘しその改良を試みる。

２．仮説推論と探索

DDS は Truth Maintenance System (TMS) (Doyle [4], McDermott [5], ATMS [1] 等）の一機能として重要な役割を果たしている。TMS は元々動的な知識の管理（無矛盾性の維持）が主要タスクであるが、DDS は問題解決のための指針を与える。TMS は仮説推論をサポートする。仮説推論は、問題解決の過程で競合する知識や不完全な知識等を取り扱う場合に必要となるものであり、それらの知識を仮説と見立ててそれに基づいた処理を行うことにより進めていく推論の形態をいう([10])。ここでは問題解決器は問題領域に依存するものとし、TMS は汎用の管理機構であるとする。ここで TMS はどのようなスタイルで実現されているかは問題とはしないが、論理ベースであることが望ましい。TMS の機能としては次の問に対して正確に答を返してくれるものを考える。

形式：D := SUPPORT(Σ, C).
入力：問題解決器から与えられた全ての式の集合Σ、
　　任意の具象式Ｃ。
問：Ｃに対してその全ての論理的サポートＤを求めよ。
答：Ｄは次の三つの性質を満足する。
　１．∀d∈Ｄに対して、Σ |= d ⊃ Ｃ.
　２．Σ∪Ｄ は 充足可能。
　３．∀d∈Ｄに対して、d'⊂d は上記１．と２．を同時には満足しない。

手続き SUPPORT に対する意味論と計算方法については [12] を参照のこと。ここで、ＤをΣに関するＣの支持仮説集合と呼び、d∈ＤをΣに関するＣの支持仮説と呼ぶ。また、Ｃをアトミックに限ると d は ATMS における環境(environment)、Ｄ は ラベル(label)に相当している。仮説推論においては、上記の Truth Maintenance 機能に対して、ある不完全知識の集合Δを取り扱うため、Ｄは Δの要素の具象例から構成されなければならない。このときΔを仮説集合としたとき、各要素の組み合わせは問題解決におけるコンテキストを決定する。多重コンテキストがある場合に、それらのどれを選択するかという問題は、現状では論理的に決定できる枠組みは存在しないし、本来問題解決において決定されるべきものであると考えられる。従って、本稿では多重コンテキストの探索に焦点を当て、DDS を多重コンテキストを管理するための汎用の TMS と、種々の推論を行うための問題領域に依存する問題解決器とのインタフェースを行うための汎用の枠組みとして位置付ける。

３．ＴＭＳのコンテキスト探索における問題点

問題解決においては、例えば設計におけるモデルの選択を考えた場合、用いる知識によって複数の競合する可能世界が考えられるため、多重コンテキストの管理が必要となる。この観点から従来の方式を考察すると、Doyle のアルゴリズム [4] は矛盾が生じたときには DDB によりそれを解消するが、バックトラッキングや仮説の選択における制御を知的に行う方法を与えていないため、多重コンテキストの問題には対処できない。従来コンテキスト機能は CONNIVER [8] のようなAI言語で実現されていたが、McDermott[5] は TMS において実現する方法を与えた。さらに ATMS [1] ではそれを明確にしており、TMS を仮説の組合せに基づいて多重コンテキストに拡張し効率を上げているが、sequential に展開する限り、やはりコンテキスト探索の制御は問題となる。ATMS においてはハッセ図の利用により environment lattice を想定

した探索を実現しており、制御として最も好ましい仮説世界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探索が期待できる。言い替えると、ATMS の探索については node の label において最簡形の environment が早く見つかるほど label の更新が少なくて済み、また nogoods の最簡形が早く見つかるほど究極的に nogood となる 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。このため ATMS ではインタフェース(consumer architecture)を通して最も仮説の数が少ない environment から問題解決を行うようにしている。探索順によりバックトラッキングは不可欠であり、de Kleer & Williams は DDB による縦型探索を ATMS が扱う environment lattice の方式へ組み込むことにより、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 [3] を提案している。しかしながら、その手法は ATMS の利点を活かしてはいるものの与えられた手続きが正確でない上に、飛躍的な効率の増加も期待できない。従って、如何なる TMS に対しても知的な制御機構の実現が課題として挙がっている。

従来の TMS が扱う DDS による他の限界としては、探索を一般の木（ＯＲ木）に限定していたことも挙げられる。例えば、SCHEMER [9] では McCarthy の AMB オペレータを実装しているが、探索木はＯＲ木として扱われている。我々は探索の背後にある論理に注目することにより、問題解決にとってより自然でＯＲ木を包含するＡＮＤ／ＯＲ木を探索の対象とするアプローチを取る。

4．コンテキスト探索アルゴリズム

ここでは Truth Maintenance 機構は前章の SUPPORT を実現できるものを想定する。多重コンテキストは基本的に AND/OR 木(あるいはグラフ) 表現が可能である。従って、AND/OR 木探索のアルゴリズム(AO*, GBF, GBB等) [6] が適用できるはずである。このとき、一つの解木（solution tree：OR の枝のうちの１つ、AND の枝は全てを持つ部分木）が一つのコンテキストに相当する。そこで、コンテキストの展開を AND/OR 木 探索に見たてて解く。ここでは、AND/OR グラフは問題解決の過程で段階的に構成されて行くものとする。即ち処理はインクリメンタルに進められるため、予め全ての仮説集合が展開されている必要はない。以下では、①仮説間に順序が与えられない、もしくは全順序が直列に与えられる場合、②仮説間にある種の優先順位が陽に（評価関数や半順序で）与えられる場合、に分けて考察する。

①【導出を用いる無評価ＡＮＤ／ＯＲ型仮説空間探索】

探索手続き(SEARCH)の概要は次の通りである。まず解くべき問題Ｐが与えられる。問題は部分問題に分割されるが、その論理的関係をΣに追加する(ADDCLAUSE)。問題解決の任意の時点で無矛盾なコンテキストは、リスト OPEN に順序付けられて格納される。１つの無矛盾なコンテキストは１つの解木に相当するが、木の葉接点のリストで表現することができる。OPEN の先頭から一つずつ解木が選択されて無矛盾性のチェック(CHECK)を行い、無矛盾であれば展開される(EXPAND)。ここで、展開とはあるコンテキストに関して、部分問題への分割、関連する問題解決手続きの実行を意味する。矛盾の生じた仮説の組合せ（ATMS の nogood に相当する）はリスト CLOSED に入れられる。尚、前述の Truth Maintenance 機能は本探索法において特にコンテキストの矛盾の発見を効率良く行えるように工夫して用いられている。そのとき、導出が別のコンテキストの絞り込みのために用いられる。

次に探索アルゴリズム SEARCH を示す。本アルゴリズムではリストとして、OPEN・CLOSED・SOLUTION・Ｄを用い、TMS の管理するデータ集合として、Σを用いる。これらを引数とした手続きも可能だが、ここではグローバル変数としておく。リストの初期状態は、OPEN については目標となる問題Ｐのみを要素として持ち、他はすべて NIL である。Ｄは各時点で無矛盾な部分コンテキストの最大集合のリストである。次の定義は幾つかのアルゴリズムのクラスとして拡張可能である。

定義 SEARCH

1．終了条件（全解探索なら、OPEN = NIL；一つの解だけでよいなら、SOLUTION ≠ NIL）を満たせば終了。解の集合として SOLUTION が得られる。
2．OPEN の先頭の要素を取り出しＬとする。ここで、L = {C, C0, C1, ..., Ck} とする。
3．P := SUPPORT(Σ, C)。
もしＰが {{}} （Σ⊢Ｃを意味する）ならばＬは、無矛盾であり、EXPAND(Ｌ)を実行して１．へ。
4．S := {{C}, {C, C0}, {C, C1}, ..., {C, Ck}, {C, C0, C1}, ..., {C, C0, C1, ..., Ck}} 。
5．S = NIL のとき、Ｌは無矛盾であり、EXPAND(Ｌ)を実行して１．へ。
6．Ｓの先頭の要素 s を取り出し、CHECK(s, Ｐ) が TRUE なら s は無矛盾であり、５．へ。また、CHECK(s, Ｐ) が FALSE ならＬは矛盾するため、ADDCLAUSE(¬s) を実行し、１．へ。

定義 CHECK(context, P)

1．context = {C, C0, C1, ..., Cn} とする。また、E := CDR(context) = {C0, C1, ..., Cn} とおく。
明らかに、∃d∈Ｄ s.t. Ｅ⊆d が成り立つ。
2．もし ∃d∈Ｄ s.t. Ｐ∪Ｅ⊆d なら return(TRUE)。
3．もし ∃d'∈CLOSED s.t. Ｐ∪Ｅ⊇d' なら return(FALSE)。
4．context に関するデモンを実行する。すべての実行に関して使われた節 p に関して、ADDCLAUSE(p) を実行する。
5．Ｄの要素でＰ∪Ｅを包含するものが無ければ、Ｐ∪ＥをＤに追加して逆にＰ∪Ｅにより包含されるＤの要素をＤから消去する。return(TRUE)。

定義 ADDCLAUSE(s)

1．s をΣに追加する。

2．Dの要素の部分集合について、s 及び s によりΣの間で導出される節と矛盾するものがあれば、それをDの要素よりすべて消去する。Dから消去されたすべての仮説集合について CLOSED に追加し、追加された要素を包含する OPEN 及び CLOSED の要素をすべて各々より消去する。return。

定義 EXPAND(L) [NB: depth-first version]

1．L := {C,C0,C1,...,Ck} とし、
CDR(L) := {C0,C1,...,Ck} とする。

2．Cが非端末ノードのとき、Cを部分問題に分割する。Cの部分問題を {CS1,CS2,...,CSm} とする。そのとき、ノードの展開に関してノード間の関係RをΣに追加（ADDCLAUSE(R)）する。
ここで部分問題がＡＮＤ関係にあれば、OPEN の先頭に cons(CS1,CDR(L)) を挿入して return。
さもなければ、すべての CSi に対して cons(CSi,CDR(L)) を OPEN の先頭に並べて挿入して return。

3．Cが端末ノードのとき、Cの最も近い祖先のＡＮＤノードで兄弟ノードが残っている節点 AS へ戻り、cons(AS,L) を OPEN の先頭へ挿入して return。
そのようなＡＮＤが存在しなければ、Lは解であり、SOLUTION に追加して return。

上のアルゴリズムに対して幾つかのコメントが必要である。SUPPORT は SEARCH のステップ３で求めているが、暗黙的に幾つかの状況で必要に応じて使用される。例えば、ADDCLAUSE のステップ２で矛盾を検出したときに、そのコンテキストを求めるときに必要である。また、CHECK はある仮説がそれを導いたコンテキストの基で矛盾しないかどうかを検出する手続きである。すなわち、負の和項 ¬A∨¬B が TMS で管理されているとき、コンテキストの探索ではこれを仮説ＡとＢが同時に存在することが矛盾すると見てその解木の枝刈りを意味し、TMS の支持仮説集合の概念を用いると仮説Ａのコンテキストの基で仮説Ｂあるいは¬Ｂの支持を問い合わせた場合、A→¬B により仮説Ｂが存在し得ないことを意味する。

SEARCH はインクリメンタルに階層化される木を探索するために次の利点を有する。まず、中間レベルでの矛盾に対応できるためにコンパイルされた知識に対応した問題解決が行える。次に、導出に対応する下位から上位への枝刈りが可能である。木上での枝刈りは１つの導出に相当し、主として CLOSED とノード間の AND/OR 関係に基づいて行われる導出である。例えば、ＡＮＤの関係にある子ノードは１つでも nogood であればその親ノードも nogood となり、ＯＲの関係にある子ノードはすべて nogood のときに親ノードが nogood となる。親ノードの枝刈りは ATMS において justification として Horn clause 以外に positive clauses を扱う場合に完全性を保証するために用いる負超導出 [2] に相当する。つまり、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」といった意味での枝刈りが導出に相当している。

SEARCH について以下の性質が成立する。（証明略）

性質1 SEARCH のステップ２で取り出されたLからCを除いたコンテキスト(CDR(L))は無矛盾である。

性質2 どの時点においても、D⊇SOLUTION。但し、Dは nonmonotonic, SOLUTION は monotonic である。

性質3 どの時点においても、∀d∈D について、∃d'⊆d で d'⊇n となる n∈CLOSED は存在しない。

定理1 最終的に求まる SOLUTION は、実際に無矛盾なコンテキストである(sound)。また全解探索のときは全ての無矛盾のコンテキストを保持する(complete)。

②【評価関数を用いるＡＮＤ／ＯＲ木探索】

仮説間に評価が与えられるときは論理の完全性は保証されないが、最良優先探索によって、組合せ的爆発を抑えることが期待できる。この場合の探索法には従来から研究されているゲーム木探索や組合せ問題に適用されている Branch and Bound（Ｂ＆Ｂ）等を①のアルゴリズム中に埋め込むことにより最適解を求めることができる。評価関数が使える時は、そのまま探索に反映できるため、DDB + environment lattice [3] 以上の効率化が可能である。例えば、ATMS のような複数の並行した仮説世界上では、単なる深さ優先探索よりも、最良優先探索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 TMS の整合ラベリングや最小集合被覆問題としての性質によるＮＰ完全性に対して、いずれのシステムにおいても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が、最良優先探索はその可能性を与えている。

・最良優先探索 (best-first search)

一般に組合せ問題に対して、深さ優先探索や幅優先探索のような方法で全解探索を行うと組み合わせ爆発が起きるため、人間の問題解決過程に見られるように、何らかの heuristics を基にして plausible path を展開していく最良優先探索を取り入れることが必要とされる。この探索法ではバックトラッキングは必ずしも矛盾に出会ったときにのみ起きるわけではない。他に良さそうな候補解がある場合の context switcing に相当するバックトラックも考えられる。従って、バックトラックする前の 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があるとすればそれは更なるコンテキストの切換えを誘引する。

・評価関数を用いた動的な枝刈り

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のための限定要因 (bound) として使用することにより、人工知能研究の基礎的な成果である B&B や α-β といった探索のアルゴリズムが利用できる。すなわち、制約式の動的な変化で不要な仮説の枝刈りを行う。例えば、ジョブショップ・スケジューリングにおける工程時間制限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、これを問題解決の過程で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。

・ヒューリスティック探索

評価関数が、解が無矛盾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。仮説選択順序のより良い指定が効率を左右する。

・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'(preference)を使う。ADB : A の方が B よりも良い（と思われる)。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ため、基本的に①の手続きで良い。より難しいのは部分的に半順序関係が与えられる場合であり、現在検討中である。

5．例題：制約満足化向きの問題解決

4.で与えたコンテキスト探索は対象とする問題領域に応じた推論戦略と合わせて適宜使って行く必要がある。一例として制約満足化（CSP）向きの問題解決器を考える。CSP は基本的に Generate and Test (G&T) によって解くことができるが、SEARCH アルゴリズムにおいて部分的に制約式を適用することで階層的に問題解決が行え、より効率的になる（hierarchical G&T)。CSP では、仮説はある変数への値の割り当てとみる。但し、中間レベルでの仮説はある中間の部分問題における状態などを表わし、それ自体は値を持たないこともある。いずれのレベルにおいても、あるコンテキストの基である観測が発見された時点でその観測あるいは否定を支持する仮説を求め、現在のコンテキストとの無矛盾性を調べる。

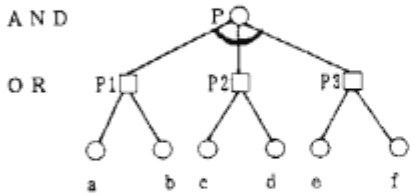
以下に [3] で使われた単純な例について考察する。

例題 問題Pは部分問題P1,P2,P3の積であり、各部分問題はパラメータであり、次の値の中から選択する。

P1∈{a,b}, P2∈{c,d}, P3∈{e,f}.

また、制約式として以下のものがある。

c∧e→false, a∧d→false, b→false.



本問題に対して総コンテキスト数は 26 個あり、ATMS では 15 個、DDB + environment lattice [3] では 14 個、本 SEARCH アルゴリズムでは 12 個のコンテキストを探索する（探索過程は右図参照)。但し本例では最下位レベルにしか制約が働かないために必ずしも SEARCH の利点は発揮されていない。

6．おわりに

本稿では、動的知識を管理するための汎用の TMS と、対象とする問題領域に依存する問題解決器との間のインタフェースとして、推論制御を行うための DDS の機能を持つ汎用の問題解決手法について述べた。その特徴としては、問題解決制御を AND/OR 木探索により行っていること、及び ATMS のようにすべての仮説を並列に扱うことをせず、コンテキストに沿ってインクリメンタルに仮説の追加が可能なことが挙げられる。この枠組みは、仮説推論システム APRICOT [11,12] への導入により実現する予定である。APRICOT は仮説推論を問題解決に使って行く場合における基本的な枠組みを提供し、論理ベースのTMS に基づいて多重コンテキストの管理を行う。課題としては、4.② のようにある種の heuristics を用いた場合に論理的な完全性と計算量の効率とのトレード・オフをどのように設定するかという点が挙げられよう。

[参考文献]

[1] de Kleer, J., An Assumption-based TMS, Artificial Intelligence 28 (1986) 127-162.

[2] de Kleer, J., Extending the ATMS, Artificial Intelligence 28 (1986) 163-196.

[3] de Kleer, J. and Williams, B.C., Back to Backtracking: Controlling the ATMS, Proc. AAAI-86 (1986) 910-917.

[4] Doyle, J., A Truth Maintenance System, Artificial Intelligence 12 (1979) 231-272.

[5] McDermott, D., Contexts and Data Dependencies: A Synthesis, IEEE Trans. Pattern Anal. Machine Intelligence, Vol.5, No.3 (1983).

[6] Pearl, J., Heuristics, Addison-Wesley (1984).

[7] Stallman, R. and Sussman, G.J., Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-aided Circuit Analysis, Artificial Intelligence 9 (1977) 135-196.

[8] Sussman, G.J., and McDermott, D., From PLANNER to CONNIVER—A Generic Approach, Proc. AFIPS FJCC (1972) 1171-1179.

[9] Zabih, R., McAllester, D. and Chapman, D., Non-Deterministic Lisp with Dependency-Directed Backtracking, Proc. AAAI-87 (1987) 59-64.

[10] 井上，仮説推論に対する一考察，ICOT Technical Memo No. TM-289, 1987年2月.

[11] 井上，導出を用いた仮説探索，情報処理学会第３５回全国大会論文集 (1987) 1567-1568.

[12] 井上，Logic-based Hypothetical Reasoning: Semantics and Problem-Solving, (to appear).

	CONTEXT	ADDCLAUSE to Σ	D	CLOSED	OPEN	SOLUTION
0		P			(P)	
1	(P)	P⊃P1∧P2∧P3			(P1)	
2	(P1)	P1⊃a∨b, A⊃a, B⊃b			(A) (B)	
3	(A)		(A)		(P2, A) (B)	
4	(P2, A)	P2⊃c∨d, C⊃c, D⊃d	(A)		(C, A) (D, A) (B)	
5	(C, A)		(A, C)		(P3, C, A) (D, A) (B)	
6	(P3, C, A)	P3⊃e∨f, E⊃e, F⊃f	(A, C)		(E, C, A) (F, C, A) (D, A) (B)	
7	(E, C, A)	¬c∨¬e	(A, C) (E)	(C, E)	(F, C, A) (D, A) (B)	
8	(F, C, A)		(A, C, F) (E)	(C, E)	(D, A) (B)	(A, C, F)
9	(D, A)	¬a∨¬d	(A, C, F) (E) (D)	(C, E) (A, D)	(B)	(A, C, F)
10	(B)	¬b	(A, C, F)	(E) (D) (B)	φ	(A, C, F)