データベース・システム　80－3
人　工　知　能　73－3
（1990．11．8）

述語論理知識ベースに適用できる高速仮説推論システム

近藤　朗子　牧野　俊朗　石塚　満
東京大学生産技術研究所

仮説推論は非単調推論系の一種であり、推論速度の遅さが実用上最大の問題点となっている。我々は、前向き推論と後向き推論の特長を取り入れることによって、従来のPrologの推論系を利用するシステムで行なわれていた無駄な探索を省き、効率的な推論を行なうことを命題論理知識に対しては可能にした。しかし、現実の問題への応用を考える場合、変数を含む述語論理知識への拡張が不可欠となる。そこで、本論文では、これらの高速化手法を述語論理知識に適用するために変更すべき点を検討し、述語論理知識ベースに適用可能な高速仮説推論の手法について述べる。

An Efficient Hypothetical Reasoning System for Knowledge-base Represented in Predicate Logic

Akiko KONDO　Toshiro MAKINO　Mitsuru ISHIZUKA
Institute of Industrial Science, University of Tokyo
7-22-1 Roppongi, Minatoku, Tokyo 106, JAPAN

A hypothetical reasoning system is an important framework toward advanced knowledge-base systems. It can be effectively applied to many practical problems including diagnosis, design, etc. However, the inference speed of its Prolog-based implementation is slow particulaly due to inefficient backtracking. In order to overcome this problem, we have developed a fast hypothetical reasoning mechanism for propositional-logic knowledge by combining the advantages of forward and backward reasonings. This fast mechanism has not been directly applicable for the hypothetical reasoning with predicate-logic knowledge. In this paper, we present a fast hypothetical reasoning mechanism for the predicate-logic knowledge as an extension of above idea.

1．はじめに

一般に人間の持つ知識の多くは、例外を持っていたり、互いに矛盾したりする不完全な知識である。不完全な知識を取り扱うことにより高次人工知能機能を実現する仮説推論は、次世代知識ベースシステムを構築する上で重要な技術であるが[1]、そのシステムは非単調推論系の一種であり、推論速度が遅いことが実用上の最大の問題点である。

推論速度を向上する手段として直接的に役立つのは、推論の筋道をガイドする役割を果たすヒューリスティックな知識の利用である。しかし、ヒューリスティックな知識はそれを獲得すること自体が困難であり、かつ問題領域全体をカバーすることができず、知識獲得のボトルネックが大きな問題となる。そこで宣言的知識表現下で仮説推論を高速化するシステマティックな方法が必要とされる。

仮説推論システムを高速化する方法として、我々は命題論理知識ベースに適用可能な、推論パスネットワークによる方法[2]や ATMS[3]のlattice構造を利用して並列に解を求める方法[4]を考案、開発してきた。いずれも仮説間の矛盾によって生じるバックトラックを回避することによって推論の効率化が達成されている。

ところが、一般的な知識は変数を用いて抽象的に表現されることが多く、あらゆる個々の具体的な事象について１つ１つ表現されているわけではない。また、逆に個々の具体的な事象をすべて表現しようとすると知識ベースが莫大になり、実用上問題がある。そのため、変数を含む述語論理知識ベースを取り扱うことは不可欠である。

そこで、本論文では変数を含む述語論理知識（正確には関数を含まない述語ホーン節）にも適用できる高速化手法について述べる。

2．論理に基づく仮説推論システム

我々の用いる論理に基づく仮説推論は Poole らが最初に Theorist[5]によって提示した枠組みに基づいており、矛盾を含む知識を仮説として取り扱うことにより不完全な知識を扱う。この仮説推論の基本機能は、ゴールとして与えられた観測をまず完全な知識のみで証明し、完全な知識だけでは証明できない場合は不完全な知識を仮説としていくつか取り入れて（これらの仮説は完全な知識と無矛盾でなければならない）、ゴールを証明することである。

仮説推論の目的は、このゴールの証明のために取り入れる仮説の集合（解仮説と呼ぶ）を求めることである。ゴールの証明は演繹推論によるが、演繹推論機能を逆向きに利用してゴールに必要な仮説を生成してくる。この解を生成する能力により、診断[6]や設計[7]等の実用的問題に適用でき、かつ発想(Abduction)にも結びつく枠組みとなっている。

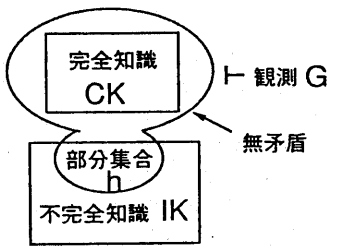
仮説推論システムの基本概念図を図１に示す。使用される知識ベースは、完全な知識（対象世界で常に成り立つ知識）の集合 CK と、不完全な知識（対象世界で常に成り立つとは限らない知識）の集合 IK の２つの部分に分かれている。システムの基本動作は、ゴールとなる観測 G に対して、

$h \subseteq IK$

$CK \cup h \vdash G$

$CK \cup h \not\vdash \square$

の３つの条件を満足する解仮説 h を求めることである。この時、解仮説 h は最小な集合であること、即ち、$h' \subset h$ で上記の条件を満たすような h' が存在しないことが望ましい。



$h \subseteq IK$
$CK \cup h \vdash G$
$CK \cup h \not\vdash \square$

図１　仮説推論システム

3．Prolog による仮説推論システムとその問題点

この仮説推論システムは Prolog の推論機構を利用すると比較的容易に実現できる。推論は Prolog の縦型探索に従って順次必要な仮説を生成・検査し、検査の結果矛盾が生じた場合はバックトラックにより仮説を入れ換えて別の仮説を生成する。そして、ゴールが矛盾なく証明された場合は、使われた仮説の集合が解仮説となる。

しかし、Prolog の推論機構をそのまま利用した単純な縦型探索では、以下に示すような無駄な探索を行うため推論の効率が悪い。図2に Prolog による推論では効率が悪い例を示す。

1) 探索する必要のない枝を探索してしまう。

ある部分木において推論パスの先が恒偽となる場合、そのパスの途中で行う推論は無駄になる。図2において、ノード j の右端の子ノードは恒偽であるため、ノード j は必ず偽となる。そのため、ノード j 以下を探索することは無駄になる。

2) 同じ枝を何度も探索してしまう。

単純なバックトラックにより仮説の変更を行う場合、変更される仮説より後の推論パス上にある部分木の探索は変更の度に何度も行うことになる。通常の証明が不能になることになるバックトラックに加えて、仮説推論では仮説間の矛盾によってもバックトラックが発生することになり、バックトラックによる非効率性が一層大になる。図2では、ノード b において、まず左端の子ノード d を選択して推論を進めた場合、仮説ノード g を探索する直前の解仮説候補は [h, i, k] である。ここで、仮説 g が加わると仮説 i と矛盾を引き起こし、バックトラックしてノード b まで戻り、次のノード候補 e を選択する。その時、ノード f に関する情報は失われてしまうので、その後の推論でもう一度ノード f 以下を探索しなければならない。

3) 同じ形の部分木を再び探索してしまう。

推論パス上に同じ形の部分木が現れても、それらを別々のものと認識し、あらためて探索してしまう。通常のバックトラックによる推論の非効率性だが、仮説推論では一層顕著になる。図2では、ノード f は2ヶ所に現れている。しかし、それらのノードを同一のものと見なすことができないので、同じ推論を繰り返すことになる。

4) 求められた解仮説が最小な集合であることが保証されない。

Prolog による推論では一度に単一の解しか求められないので、推論の途中でその解が最小であるとは確認できない。

例えば図2では、ノード d のサポートする仮説は[h, i] 、ノード e のサポートする仮説は[i]である。ノード d はノード e に対し冗長なため、その後の推論を行う必要はないのだがしてしまう。

これらの問題を解決するような我々が開発した高速仮説推論システムについて4節で述べる。

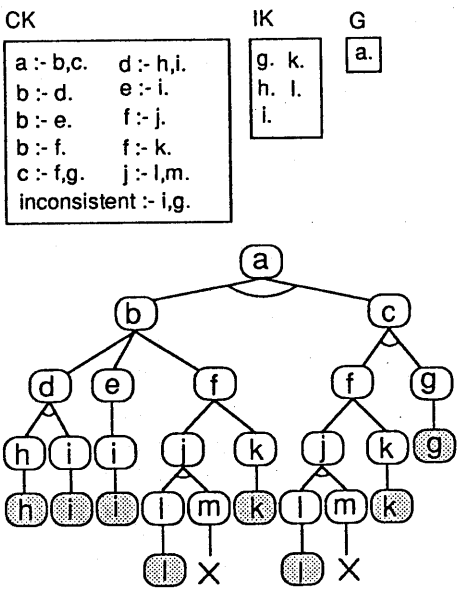


図2　Prolog で無駄な探索をする例

4．従来の高速仮説推論システム

推論を行う方法として、後向き（トップダウン）推論と前向き（ボトムアップ）推論の２通りが考えられる。

Prolog に代表される後向き推論では、ゴールから出発するため問題に関係のないノードを探索することはないが、同じノードを何度も探索する可能性がある。特に仮説推論の場合は、仮説間の矛盾のためバックトラックを頻繁に繰り返し、無駄な探索をする可能性が高い。

一方、全解探索的な前向き推論では、同じノードを繰り返し生成することはないがゴールに無関係なノードをも生成してしまう。ATMS[3]は前向き推論のプロダクションシステムと組み合せ、ゴールに向かう推論の筋道をユーザの記述するルールによって制御し、そのもとで無矛盾な仮説の集合（環境）を見い出す。

３節で述べた問題点は部分的に前向き推論を取り入れることにより解決する。

問題点 1）は、推論の途中で枝別れがあった場合 Prolog が単純に左端ノードから深さ優先の縦型探索を行うことに起因する。即ち、右側のノードに恒偽のノードがあってもそれに気づくことはできない。仮説推論の場合、仮説の合成の処理に非常に時間がかかるので、そのような処理をする前に不要な枝（恒偽のノードを含む推論パス）を刈ることが重要である。これは推論パスをつける後向きのフェーズと仮説を合成する前向きのフェーズとを分けることにより解決する。

問題点 2）は、Prolog が単一の環境しか取り扱えない点に起因する。これは前向き推論を行う際に複数環境を取り扱うことにより、仮説間の矛盾によって生じるバックトラックを避けることができるので解決できる。

問題点 3）は、ネットワーク化して同じノードを１つにまとめること（一種のコンパイル）により解決できる。

問題点 4）は、2）同様に前向き推論を行う際に複数環境を取り扱えば、ノード毎に冗長な仮説の組み合せを排除することができ、早めに解仮説を絞り込むことができる。

以上のような観点に基づき、我々は命題論理知識ベースに適用可能な高速仮説推論システムとして次の２つのシステムを開発した。

① 推論パスネットワークによる方法[2]

はじめに、知識ベースに基づいてノード間にリンクを張り、観測事象をゴールとするゴール駆動で真理値を伝播させて、ゴールに到る推論パスネットワークを完成させる。この時、恒真、恒偽となるようなパス、すなわち仮説の合成に寄与しないパスは刈ってしまう（パス生成フェーズ）。この段階で、問題点 1),3)は解決される。この推論パスネットワークの形成は Dowling & Gallier の命題ホーン節に対する線形時間推論アルゴリズム[8]に、基づいている。

次に、この推論パスネットワークに沿って仮説を伝播させ、ノード毎に伝播してきた仮説を合成

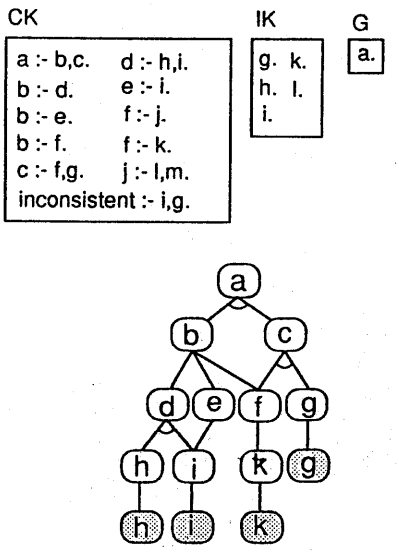


図３　推論パスネットワークによる推論例

する（仮説合成フェーズ）。この場合仮説の合成はATMS[3]で採られている方法と同じように複数環境を取り扱っているので、問題点 2),4) は解決される。

例えば、図2の例は①の高速仮説推論システムを利用した場合、図3に示すような構造の推論パスネットワークを利用して仮説を合成すればよく、効率的な推論を行うことができる。

② ATMS[3] の lattice 構造を利用して並列に解を求める方法[4]

はじめに、観測を後向き推論により仮説のみを含むサブゴールを残して展開（unfolding）する。この展開により、恒真、恒偽となるようなノードは吸収されてしまい問題点 1) は解決される。

次に残された各サブゴールに対して仮説を Environment lattice 上にマッピングし、最後にそれらを合成する。サブゴールが複数環境を保持しているので、問題点 2),4) は解決される。

尚、このシステムは知識のネットワーク構造へのコンパイルを行なわないため、問題点 3) は解決されていない。

これらのシステムは命題論理知識ベースに対して極めて大きな速度向上を達成した。特に①のシステムは従来の Prolog の推論機能を利用したインプリメントと比較して、1000 倍以上の高速推論を可能にした。

5．述語論理知識への適用

前節の高速仮説推論法を述語論理知識へ適用することを考える。述語論理ではノード間の変数のバインディングの問題があり、従来の命題論理知識に対する高速仮説推論法をそのまま述語論理知識に適用することはできない。

まず、① の推論パスネットワークによる方法[2]について考える。例えば、述語論理知識に対する初期ネットワークは図4のように表される。ここで、各ノードにおいて変数の付け直しを行っている。それは、述語論理知識ベースにおいて各述

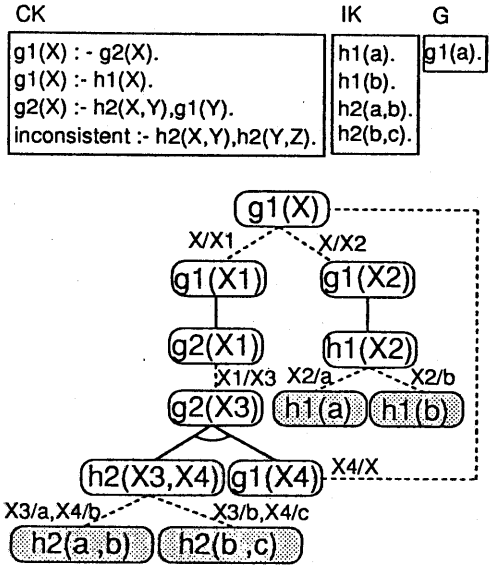


図4　述語知識に対する初期ネットワーク

語知識間における変数は独立であるからである。

ここで、観測を g1(a) としてネットワーク上で変数の値を伝播させると、X/X1/X3/a, X4/X/b となり、X/a, X/b という矛盾が生じる。これは g1(a), g1(b) が全く別のノードであるにもかかわらず、ネットワーク上で同じ g1(X) と表してしまうところに問題がある。一般に、述語知識というのは変数の値が決定されて初めて実体としての意味を持つ。そこで、推論パスネットワークを作成するには述語知識をエルブラン領域に展開する必要がある。しかし、その場合ネットワークの規模が大きくなりすぎるので実用的ではない。

そこで、②の ATMS[3]の lattice 構造を利用して並列に解を求める方法[4]をベースにして、述語論理への拡張を考える。ただし、以下に示すような理由により、展開(unfolding)という操作は行わない。

(1) 仮説を用いずに展開すると非効率な場合がある。

仮説を用いないと完全に変数を定数化できない

ので、ノード間でユニファイを行う際に変数が残ったままでバインドされてしまう。そのため、探索する必要のない枝も探索する可能性がある。例えば、図5の例の場合、仮説 h1(a),h2(b) を用いれば g3(X1,X) は g3(b,a) と決定するので、その後は g3(b,a) をゴールとする推論を行えばよい。しかし、仮説を用いない場合は変数 X,X1 の値が決定しないので g3(X1,X) をゴールとする探索を行わなければならず、無駄な探索をしてしまう。この例では本来は仮説 h4(a) だけを探索すればよいところを、h3(a),h3(b),h4(a),h4(b) すべて探索してしまう。

(ロ) 仮説を用いずに展開すると展開できない場合がある。

(イ) と同じ理由による。再帰構造をとる述語知識がある場合、変数の値が決定されていないと、いつまで展開しても止まらないことがある。命題論理の場合は、再帰的になる命題知識があるとそれは知識ベースそのものに欠陥があるということになる。しかし、述語論理の場合は、構造が再帰型でも　変数が決定されれば論理的には再帰的にならないことがあるので、再帰構造をとる述語知識を含む知識ベースというものは存在する。図6はその例である。この例でも、仮説を用いることにより全ての変数の値が決定されれば探索は止まり、最終的な解を求めることができる。

以上のことから、述語論理知識に対しては仮説も用いてノードの変数の値を決定し、その値をノード間で伝播させることが必要となる。そこで、我々は”各ノード毎に後向きで推論すべきノードを決定し、それらの子ノードがすべて確定したら前向きで仮説の合成をする”という方法を基本方針として、述語ホーン節知識（関数を含まず）に適応できる高速仮説推論システム KICK-HOPE (Knowledge-Base Handling Incomplete Knowledge - by Holding Parallel Solusion on Environment Lattice) を開発した。尚、この手法は演繹データベースの問い合わせ処理における QSQR 法の考え方[9]と関係しているが、矛盾の可能性をもつ仮説を扱う点に特徴がある。

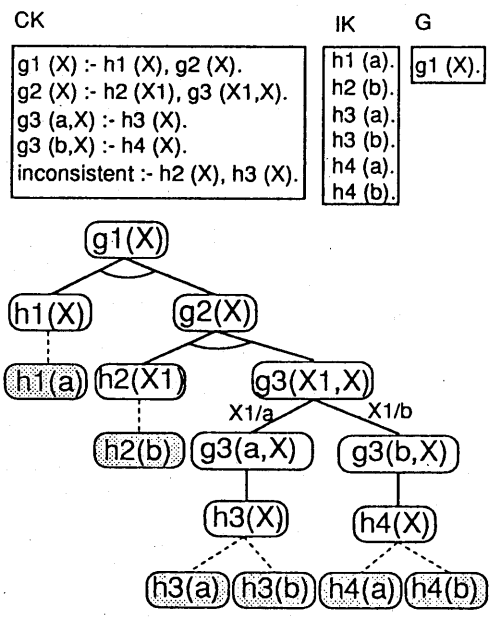


図5　仮説なしで展開すると非効率な例

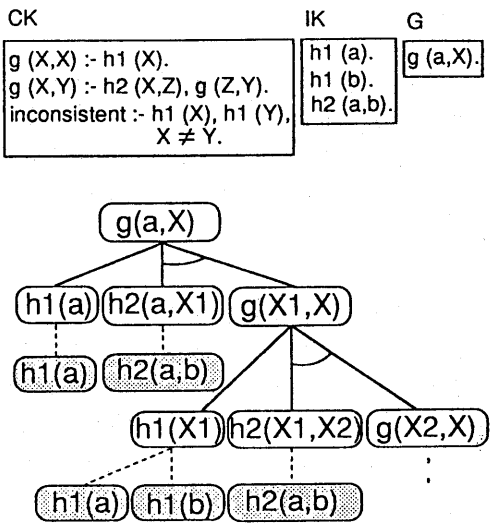


図6　仮説なしで展開できない例

6．述語ホーン節知識を扱う高速仮説推論システム KICK-HOPE

仮説は一般にルール型であってもよいが、命題ホーン節の高速仮説推論法[2,4]と同様に、仮説となる単節を導入してルール型の仮説をなくし、仮説はすべて単節（演繹データベースにおける外部データベース(IDB)に置きかえる。これによってすべての仮説は推論木のリーフに位置するようになる。

KICK-HOPE の取り扱うノードのデータ構造は

＜ノード名、サポートする解仮説＞

となっている。初期状態では、ノード名は変数を含んでいても構わず、サポートする解仮説は未定である。ノードを KICK-HOPE の推論器にかけると、確定ノードが生成される（図7参照）。このように、あるノードに対するすべての確定ノードを生成することを「ノードを解く」と表現する。

ここで、確定ノードとはノード名に変数を含まず、次のいずれかと決定されたノードである。

1. サポートする仮説が不要である。
   → true ノード
   ＜ノード名、true ＞
2. 失敗する。→ false ノード
   ＜ノード名、false ＞
3. サポートする解仮説が決まる。
   ＜ノード名、サポートする解仮説＞

KICK-HOPE の推論器の構成を図8に示す。

あるノードを推論器にかけると、まずそのノードが and ノード、or ノード、それ以外のいずれであるかを判定し、and ノードならば and ノード処理、or ノードならば or ノード処理、それ以外ならば知識ベースサーチ処理をそれぞれ行う。

and ノード処理、or ノード処理のアルゴリズムを以下に示す。

<< ノード A and B に対するアルゴリズム >>

① ノード A を解く。（1つまたは複数の確定ノードが生成される。）

② ノード A より生成されたすべての確定ノードをノード B にユニファイさせる。

③ ユニファイされた1つまたは複数の B のノードをそれぞれ解く。

④ ノード A,B の互いにバインドする確定ノード間でサポートする仮説を合成する。この時無矛盾性チェックを行い、矛盾する仮説の組を除く。

<< ノード A or B に対するアルゴリズム >>

① ノード A,B をそれぞれ解く。（ノード A, B ともに確定ノードが生成される。）

② 冗長な仮説の組を除く。

知識ベースサーチ処理は上記の and ノード、or ノード以外、すなわち、ノードが単節(unit clause)である場合に行う。

<< ノード A に対する知識ベースサーチ処理 >>

ノード A にユニファイ可能なすべての知識に対して、それぞれ返すノードを求め、リストにして返す。

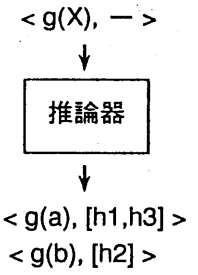


図7　確定ノードの生成

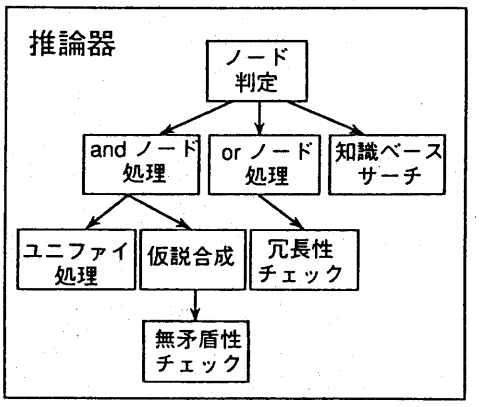


図8　KICK-HOPE の推論器の構成

返すノードはユニファイされる知識の種類により次の４つに分類される。

・ルール型の完全知識のヘッド部とユニファイされる場合：

その知識のボディ部を返す。これはまだ確定ノードではないので、このノードはさらに解くことになる。

・事実型の完全知識とユニファイされる場合：

true ノード（確定ノード）を返す。

・不完全知識とユニファイされる場合：

その知識をサポートする解仮説とする確定ノードを返す。

・ユニファイされる知識がない場合：

false ノード（確定ノード）を返す。

KICK-HOPE のような並列に解を保持していくシステムでは、仮説の合成や冗長な仮説の削除といったマージの処理に時間がかかる。しかし、無矛盾性チェックや冗長性チェックは、すでにノードが確定した後に行われるので、命題論理の場合と同じく Environment lattice 上でのビット演算として高速計算することができる。

７．例題による推論速度の比較

図９、図１０に示す２つの例題（例１、例２）により、従来の Prolog によるシステムと KICK-HOPE とで推論速度の比較を行う。

これら２つの例は完全知識（CK）は全く同じで、不完全知識（IK）は例２が例１から h(X,1) という知識を除いたものとなっている。知識の規模を仮説知識 h(X,Y)[X = 1,2,...,n-1, Y = 1,2,3] の n で表すと、いずれも展開されるノード数は 6n+4 で表される。例１は Prolog では１つめの矛盾知識（inconsistent :- h(X,1) & h(X,3).）によるバックトラックのため同じノードを何度も探索してしまう効率の悪い例である。例２はバックトラックによる効率の悪さが全く起こらない例である。

図１１の例１の結果より、Prolog によるシステムは知識の規模の増大とともに急激に推論時間がかかっていることがわかる。例えば、n = 3 では KICK-HOPE が 0.12 sec 、Prolog が 0.13 sec であるが、n = 15 では KICK-HOPE が 2.22 sec 、

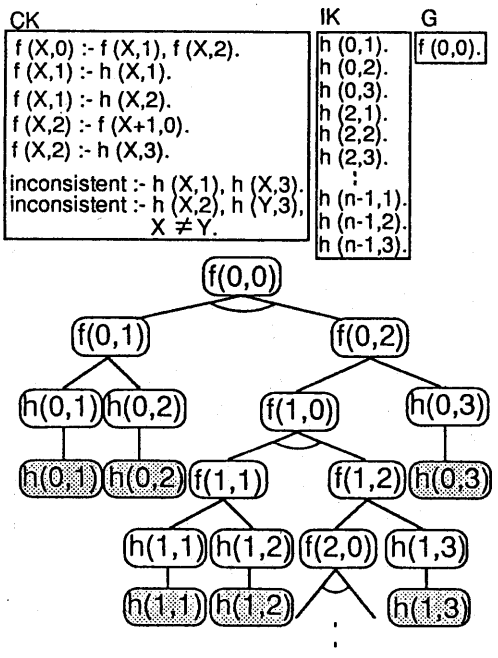


図９　例１の知識ベースと推論木構造

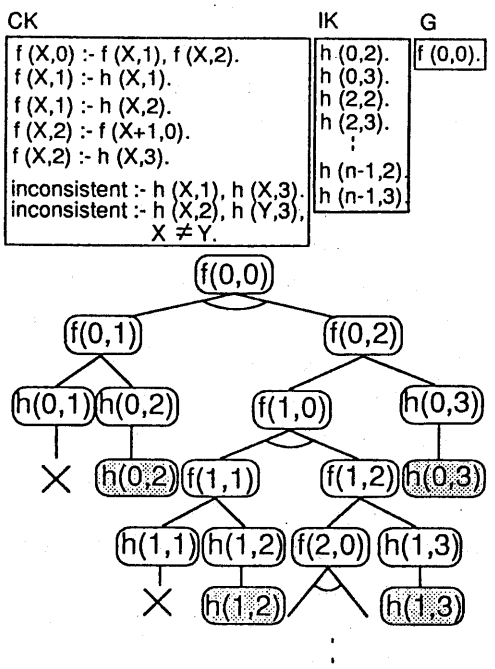


図１０　例２の知識ベースと推論木構造

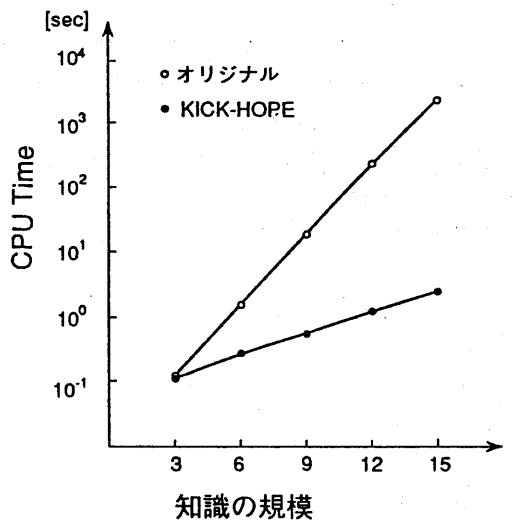


図１１　例１の知識ベースと
ゴールに対する推論時間

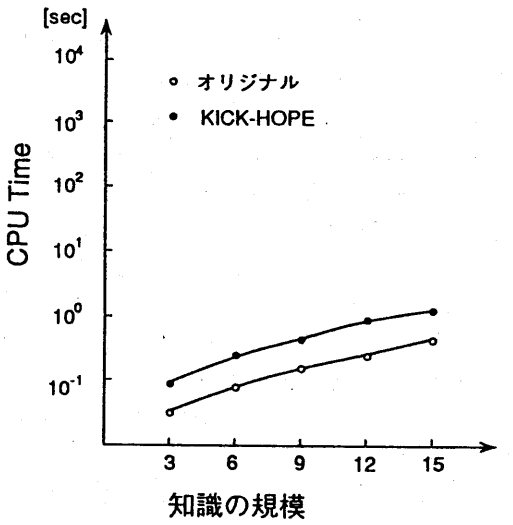


図１２　例２の知識ベースと
ゴールに対する推論時間

Prolog が 2693.86 sec となっている（いずれも SUN-4 上での時間）。

一方、図１２の例２の結果では逆に Prolog によるシステムの方が KICK-HOPE よりも高速である。これは例２ではバックトラックによる探索の非効率がないので、KICK-HOPE はマージにかかる時間の分だけ遅くなっているものと考えられる。しかし、この時間の増加は知識規模に対してゆるやかな上昇であり、かつ Prolog によるシステムのほぼ定数倍（約３倍）に留まっている。

２つのグラフを比較すると、Prolog によるシステムはノード数が同じでもバックトラックいかんにより、推論速度にかなりの差がでるが、KICK-HOPE では推論速度はノード数にのみ依存するということが分かる。一般の知識ベースではこれらの例の中間、つまり適度にバックトラックが含まれるということが予想されるので、推論速度の点で KICK-HOPE の大きな優位性が現れる。

８．まとめ

変数を含む述語ホーン知識（関数を含まない）に対する高速仮説推論版である KICK-HOPE について述べた。KICK-HOPE は Prolog の推論機能を利用する仮説推論のインプリメントにおける３節に記した４つの問題点のうち、2),4) の２つの問題点を解決した。

2) の同じ枝をを何度も探索してしまう問題点はバックトラックによる Prolog の推論の最大の欠点ともいえるもので、仮説推論では仮説間の矛盾によってもバックトラックが生ずるので、非効率性は増大する。この問題を解決したことにより推論速度は飛躍的に向上した（７節参照）。

問題点 4) に対する解決策として冗長な解仮説をノード毎に取り除くということは、述語論理知識を取り扱う場合かなり重要である。KICK-HOPE では、左のノードを確定して右のノードにユニファイさせている。この時、左のノードにおいて冗長なノードを排除することは、ユニファイするノードを少なくすることになるので、右側の探索空

間を狭めることになる。

このように KICK-HOPE は、探索木のうち最終的に解仮説を求めるのに必要な推論パスのみを１度だけ探索する。しかし、一般に探索ノードの数自体が知識の規模に対し指数オーダーで増大するので、KICK-HOPE の高速推論も図１１、１２に見られるように、指数オーダーの壁を超えることはできない。これ以上の高速化は仮説合成などのマージのコストを下げるなどテクニック的な要素が重要となる。これらの手法に関しては、関係データベースの分野で演算 join をビットベクトルを用いて効率的に行う方法[10,11]がある。（仮説推論を含む非単調推論の計算複雑度は NP-完全、あるいは NP-困難と証明されており[12]、変数を含まない命題論理においても通常の意味の解の探索では指数オーダーの壁を超えることはできない。）

指数オーダーの推論時間の壁を超えるには、知識自体の変換（学習）[13]や過去の事例の利用（類推）[14] を行う必要があり、研究を進めている。

＜参考文献＞

[1] 石塚：不完全な知識の操作による次世代知識ベースシステムへのアプローチ、人工知能学会誌、vol.3, No.5, pp.552-562 (1988)

[2] 伊藤、石塚：論理制約利用による高速仮説推論システム、情処学会人工知能研資料、70-5 (1990.5)

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

[4] 近藤、牧野、石塚： Lattice 構造を使った並列横型解法による仮説推論システムの高速化、情処学会第40回全大、6C-1,(1990.3)

[5] D.Poole, R.Aleliunas and R.Goebel: Theorist; A Logical Reasoning System for Defaults and Diagnosis, in The Knowledge Frontier : Essays in the Knowledge Representation (N.J.Cercone and G.McCalla (eds.)), Springer-Verlag, N.Y (1987)

[6] 松田、石塚：仮説推論システムの拡張知識表現と概念学習機構、人工知能学会誌、Vol.3, No.1, pp.94-102 (1988)

[7] 牧野、石塚：制約評価機構付き仮説推論システムとその回路ブロック設計への応用、人工知能学会誌、Vol.5, No.5, pp.640-648 (1990)

[8] W.F.Dowling, J.H.Gallier : Linear-time Algorithm for Testing the Satisfiability of Propositional Hoan Formulae, J. of Logic Programming, Vol.3, pp.267-284, (1984)

[9] 西尾、楠見：演繹データベースにおける再帰的な問い合わせの評価法、情報処理、Vol.29, No.3, pp.240-255 (1988)

[10] Stanley.Y.W.Su : Database Computers, McGrow-Hill (1988)

[11] Esen Ozkarahan : Database Machines and Database Management, Prentice-Hall (1986)

[12] H.A.Kautz, B.Selman : Hard Problems for Simple Default Logics, Proc, KR'89, Tront, pp.189-197 (1989)

[13] 牧野、石塚：経験に基づく学習機能を備えた仮説推論システム、1990年信学会秋期全大、D-155, (1990)

[14] 阿部、石塚：推論パスネットワーク上での類推による高速仮説推論システム、情処学会人工知能研資料、72-2 (1990.9)