人　工　知　能　82－5
（１９９２．　５．　２６）

述語論理知識を扱う仮説推論の高速最適解推論法

近藤　朗子　　石塚　満
東京大学生産技術研究所

不完全な知識を扱う仮説推論は非単調推論系の一種であり、低い推論速度が実用上最大の問題点となっている。我々はこれまでに述語論理知識を扱う高速仮説推論システムとしてKICK-HOPEを開発した。このシステムは全解探索をするシステムであるため、特に仮説に対する制約が緩い場合、解となる仮説の組合せ数が膨大となり、仮説合成の計算に計算時間がかかるという問題点があった。一方、現実には全ての解を得るよりも、より望ましい解を高速に得る方が役に立つ場合が多い。そこで、KICK-HOPEの高速推論法をベースにして最適解を効率よく求める方法を考案し、実装したシステムの推論速度の向上効果を例題によって検証した。

An Efficient Inference Method for Obtaining the Optimal Solution in Predicate-Logic Hypothetical Reasoning

Akiko KONDO　and　Mitsuru ISHIZUKA

Institute of Industrial Science, University of Tokyo
7-22-1 Roppongi, Minatoku, Tokyo 106, JAPAN

A hypothetical reasoning is an important framework toward advanced knowledge-based systems. The most crucial problem with the system is its slow inference-speed due to its non-monotonic inference nature. We have developed a fast hypothetical reasoning system with predicate Horn clause expression to overcome this problem. However, when constraints for the hypotheses are not so strong, the number of synthesized hypotheses becomes too large to calculate. We present an efficient hypothetical reasoning method for obtaining the optimal solution, which is the most desirable solution in many cases. The effectiveness of this method is shown experimentally by the faults diagnosis problems of logic circuits.

1．まえがき

一般に人間の知識は、例外や矛盾を伴っていたり、部分的に欠落している不完全な知識である。知識ベースシステム上で高次人工知能機能を実現するためには、このような不完全な知識を取り扱えるようにすることが重要である。仮説推論は、そのような不完全な知識を仮説として取り扱うことができる[1)]。

一般に不完全な知識を取り扱う推論は非単調推論となるため、推論速度が遅いという問題点があり、仮説推論においても推論速度を向上させることが重要な課題となっている[2),3),4)]。

我々は、述語ホーン節知識を扱う高速仮説推論システムとして KICK-HOPE というシステムを開発した[4)]。これは演繹データベースの QSQR 法（Query / Subquery method）[5)]を援用し、同じ推論木での再計算を回避することにより効率化を達成する高速仮説推論システムとなっている。また、変数を含むホーン節を扱うことから、知識表現力も高い。

しかし、この KICK-HOPE は全解探索をするシステムであるため、特に仮説に対する制約が緩い場合、解となる仮説の組合せの数が膨大となり、仮説合成の計算に非常に時間がかかるという問題点があった。一方、現実的には全ての解を得るよりも、より望ましい解を高速に得る方が役に立つ場合が多い。

そこで、本論文では KICK-HOPE の高速推論法をベースにして最適な解を（少なくとも１つ）効率的に求める方法について述べ、実装したシステムの推論速度の向上効果を例題によって検証した。

2．論理に基づく仮説推論システム

一般に仮説推論とは、真か偽か不明な事項をとりあえず真として（仮説を立てて）推論を進め、うまくゴールに到達したら立てた仮説は正しかったとみなすという推論の形式である。我々の用いる論理に基づく仮説推論は Poole らが Theorist[6)]によって提唱した枠組みに基づいており、他と矛盾の可能性を有する不完全な知識を仮説として取り扱う。

この仮説推論の基本機能は、問題となるゴールをまず完全な知識（常に真の知識）のみで証明することを試み、完全な知識だけでは証明できない場合は不完全な知識である仮説を利用して、ゴールを証明することである。ただし、この証明の際に利用した仮説同士は無矛盾でなければならない。

仮説推論の目的は、ゴールを証明すること自体よりむしろ、ゴールの証明のために必要な互いに矛盾しない仮説の組（解仮説と呼ぶ）を得ることである。ゴールの証明のための演繹推論機能を逆向きに利用し、ゴールの証明に必要な仮説が生成する。この仮説を生成する能力により、診断[7)]や設計[8)] 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。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 （Complete Knowledge）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（Incomplete Knowledge）の２つの部分に分かれている。システムの基本動作は、与えられたゴール Gが完全な知識のみで証明できない場合、

$h \subseteq IK$　　（h は IK の部分集合）

$CK \cup h \vdash G$　（CK ∪ h から G が証明される）

$CK \cup h \not\vdash \square$　（CK ∪ h は無矛盾、□：空節を表す）

の３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、

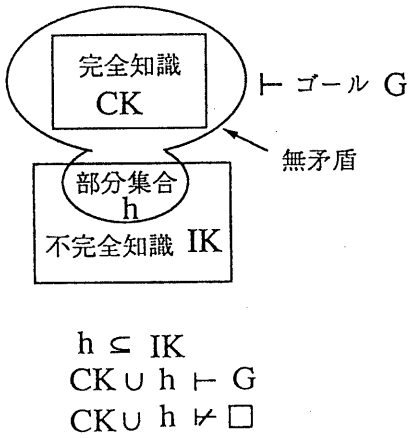


図１　仮説推論システム

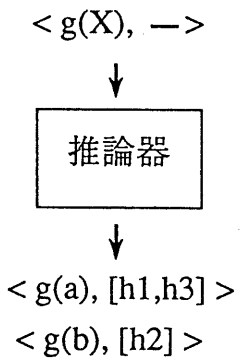


図２　KICK-HOPE の推論例

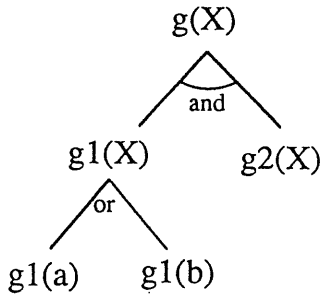


図３　推論木１

即ち、ｋ⊂ｈ で上記の条件を満たすような ｋ が存在しないことが望ましい。

一般に解仮説 h は複数個存在する。全解探索をするということは、すべての解仮説を求めることに相当する。本論文で述べる最適解探索をするということは、仮説に何らかの重みを付け、それに基づいて計算された評価値が最小（あるいは最大）の解仮説を少なくとも１つ求めるということに相当する。

３．述語論理知識を扱う高速仮説推論システム

先に述べたように、仮説推論システムの最大の問題点は推論時間が十分でないという点であった。

問題領域に依存しないシステマティックな高速推論法としては、不必要な推論を行わない、即ち、同じ推論を繰り返したり、ゴールの証明に寄与しない推論を避けるということが重要となる。　それは、前向き推論と後向き推論を適当に組み合わせることによって実現でき、我々もその考え方に基づく高速仮説推論システムを幾つか開発してきた[2),3),4)]。

ここでは、それらの例のうち本論文で述べる推論法のベースとなった述語ホーン節知識を扱う高速仮説推論システム KICK-HOPE (Knowledge-base Handling Incomplete Knowledge - by Holding Pararell Solusion on Environment Lattice)[4)] について簡単に述べておく。

KICK-HOPE の推論法の基本は、図２の例に示すように問題となるノードについて、そのノードの取り得る変数値と、そのノードを証明するために必要な仮説（サポートする仮説）を得ることである。このことを確定ノードを生成するという。これをゴールを証明するのに必要なすべてのノードについて行う。

あるゴールを証明するためにどのノードが必要であるかは後向き推論により決定する。図３の例では g(X) を証明するには g1(X) と g2(X) が必要であるというような情報である。この時、確定ノードを１つ生成する毎にそのノードの変数の束縛情報を他のノードに伝播させることによって、無駄なノードを生成させるのを抑えている。この例では g1(X)　の確定ノードが生成された時点で、X の取り得る値が a or b と決定されるため、g2(X) の X の値も a or b と制限される。この機能は再帰的な述語のルールが知識として与えられた場合に推論が無限ループに陥るのを防ぐ役割も果している。

また、確定ノードの合成の際に行われる仮説の合成は、非効率的なバックトラックを行わない前向き推論で行う。図３の例では、g(X) をサポートする仮説は g1(X) をサポートする仮説と g2(X) をサポートする仮説とをマージして得る。この KICK-HOPE は全解探索をするシステムであるので、一般に確定ノードをサポートする仮説を保持するには多重環境が必要となる。これについては、ATMS[9)] で利用されているの lattice 構造の考え方に基づき仮説の合成を行い、最小の仮説の集合 (minimal set) を得ている。

この KICK-HOPE の推論法の特長は、前向き推論と後向き推論を組合せることによりゴールを証明するのに必要なノードのみを１度だけ推論するという点である。その結果、単純な Prolog の推論機能を利用した仮説推論システムと比較して大幅な高速推論を達成した[4]。しかし、KICK-HOPE は全解探索をするシステムであるため、特に仮説に対する制約が緩い場合、サポートする仮説の数が膨大になり仮説合成の組合せ計算（マージ）に非常に時間がかかるという問題点があった。そこで、我々は全解を求めるのではなく、最適な解仮説を得るということを目的として、仮説合成の組合せ計算を減らすことにより高速に解を得る方法を開発した。次節以降にその手法について述べる。

４．最適解の推論方法

本節では、３節で述べた KICK-HOPE の効率的な推論の枠組みを利用して最適解を求める方法について述べる。KICK-HOPE では、図２に示すように、確定ノードにおいて、サポートする仮説の組をすべて求めている。しかし、最適解を求めるのであれば、確定ノードにおいて、すべてのサポートする仮説の組を求める必要はない。そこで、最適解推論においては、確定ノードの生成に際して最適な仮説の組をとりあえず（少なくとも）１つ求めることにする。その後の推論で、その最適な仮説の組が不適当であると判断された場合（他の仮説との合成で矛盾を生じた場合、あるいは、残りの仮説の組合せの中に最適解が存在する可能性が生じた場合）は、次の解仮説の候補を求める。この方法により、極力合成する仮説の数を減らし、高速に最適解を得ることができる。

最適解の基準として、各要素仮説に重みというものを与え、その要素仮説の重みの和を最小とする最適解を求める。以下に、仮説の重みと最適解を高速に得るための仮説の合成方法について詳しく述べる。

４．１ 仮説の重みと未合成仮説群の下界値

最適界を求める基準として仮説の重みを利用する。重みは１以上の整数値とし、仮説の組の重みは、その要素仮説の重みの和とする。最終的なゴールを証明するのに必要な仮説の組の重みが最小であるような仮説の組合せを最適解とする。その

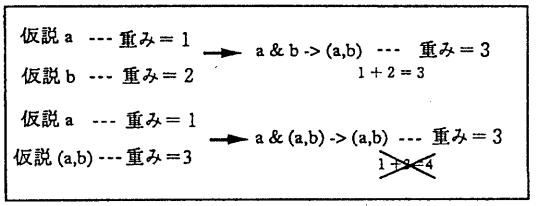


図４　仮説の重みの計算

意味から、重みが小さいほどその仮説の優先度は高くなる。要素仮説の重みが設定されていない場合は、重み＝１（デフォルト値）とする。この場合は、組み合わされた仮説の数が多いほど重みが大きくなるので、仮説の数が少ないものの優先度が高くなる。

ここで、４．２項で考える仮説の合成に際し、どの組合せから計算するかという情報を得るために、未合成仮説群（まだ仮説の組合せ計算をしていない仮説のグループ）の下界値を求めておく。未合成仮説群の下界値とは、その群の中の仮説の組合せの重みは必ずその値以上になるという情報である。未合成仮説群の下界値は以下のように求める。

and ノード ... 子ノードの下界値の最大値を and ノードの下界値とする。
or ノード ... 子ノードの下界値の最小値を or ノードの下界値とする。

and ノードにおいて、子ノードの下界値の和が and ノードの下界値になるのではなく、子ノードの最大値が and ノードの下界値となる理由を説明する。例えば、仮説 a の重みが 1、仮説 b の重みが 2 である時、仮説 (a, b) の重みは a と b の重みの和 1+2=3 となる。ところが、and ノード a & (a, b) の合成結果は (a, b) となり、重みは 3 で、1+3=4 とはならない（図４参照）。即ち、仮説の and 演算では、一方の仮説が他の仮説に包含されたり、共通項を持つ場合には、重みは単純な和とはならない。しかし、重みが最も

小さくなるのは、一方が他方に包含される場合であるので、下界値は子ノードの下界値の最大値となる。

一方、or ノードにおいて、子ノードの下界値の最小値が or ノードの下界値となるのは明らかであろう。

（付随的な利点として、一般に多重環境（複数の仮説の組合せ）を取り扱う場合、冗長な仮説の組を取り除く必要があるが、仮説の重みを１以上の整数と設定した場合、最適解は他のものに対して冗長となることはないので、冗長性のチェックをする必要がなくなる。）

４．２ 高速に最適解を得るための仮説の合成方法

本項では、４．１項で考えた仮説の重みと未合成仮説群の下界値を利用して、高速に最適解を得るための仮説の合成方法について記す。

仮説合成の回数を極力減らすために、我々は次の３種の考え方を用いる。

(a) ビーム探索法
(b) 最良優先探索法
(c) 分枝限定法

以下に、これらの考え方を簡単に説明し、図５の例題を用いて仮説推論への適用を示す。

(a) ビーム探索法[10)]

この方法は、探索の途中で評価値を利用して複数ある候補から解を絞る方法である。それにより、

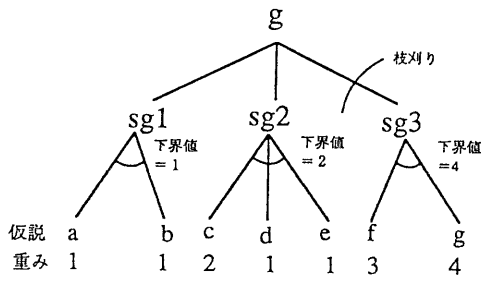


図５　推論木２

解の組合せの個数が増大するのを防ぐ。解の選択の方法としては次の２通りがある。即ち、評価値の良い順に一定の個数（ビーム幅）だけを残し、残りの候補は切り捨てる方法と、一定のしきい値を設け、評価値がそれを超えるような候補は切り捨てる方法とがある。

前者はビーム幅の設定が難しい。ビーム幅が大きいと解の個数が多くなるため、仮説の組合せ数が増大し効率化は難しい。しかし、ビーム幅が小さいと削除されたものの中に解が含まれている可能性が高くなる。特に、仮説推論の場合は残された仮説がすべて後の推論で矛盾となってしまうというような可能性も高い。一方、後者はこれ以上評価値の悪いものは残しておいても仕方がないという場合に有効である。そのような値をしきい値として設定しておけば、グローバルな制約として利用することができる。

我々は後者を採用した。図５の例では、例えばしきい値を 3 とすると、サブゴール sg3 の下界値は 4 なので、サブゴール sg3 は取り除かれることになる。

尚、ビーム探索法では必ずしも最適解が得られるとは保証されていない。

(b) 最良優先探索法[10)]

この方法はその時点で最も良い評価値の順に探索する方法であるが、我々はこの考え方を探索にではなく、仮説の合成の順序を決めるために利用する。

一般に評価値を得るためにはヒューリスティクスが必要となるが、４．１項で述べた方法により、仮説合成をする前に容易に下界値を得ることができるので、その下界値を評価値とする。また、最良優先探索法では、評価値が下界値となる場合は必ず最適解が得られることが保証されているので、本システムでも最適解が得られることが保証される。ところで、最良優先探索法は効率的に最適解を得る代償として探索過程での記憶量が増大するという欠点を持つ。しかし、我々は、(a) のビーム探索法を併用することにより、解となる可能性のない仮説を削除し、その欠点を転減している。

図５の例では、サブゴール sg1 の評価値は 1、サブゴール sg2 の評価値は 3 であるから、サブゴール sg1 から先に仮説の合成を行う。ただし、

合成の途中でサブゴール sg2 の評価値 2 を超えた場合はその合成を途中で保留にし、サブゴール sg2 の仮説の合成を開始する。

(c) 分枝限定法[11]

この方法は、解をとりあえず1つ生成し、その評価値を暫定値として憶えておき、以後の推論でその値よりも評価値が良くなる可能性のない枝を刈る。途中で評価値がさらに良くなった場合は暫定値を書き換えていくという方法である。

図5の例では、サブゴール sg1 の最適解仮説は (a,b) で重みは 2 である。そこで、暫定値を 2 とすると、サブゴール sg2 の仮説合成をする途中で、 sg2 の最適解仮説の重みが 2 より大きくなる、即ち、仮説 c と仮説 d を合成した時点で重みは 3 となるので、 これ以上、仮説の合成を続ける必要はなくなる。

我々は、この方法を、まだ確定していないノード間で利用する。即ち、(a),(b) の方法は各ノード内での仮説の合成に利用し、確定ノードとして最適解を生成していたが、今度は1つの確定ノードで最適解が得られた時、その重みを他のノードに伝播することに分枝限定法を利用する。それによって、すべてのノードにおいて最適解仮説を求める必要はなくなる。

一般には、分岐限定法は最終ゴールに対して行う。なぜなら、中間ノードで最適解を得ても、それが最適解の一部になるとは限らない。また、後の推論でその解が矛盾となってしまう場合も生じる。そのため、中間ノードでは最適解仮説の他にまだ計算されていない仮説の組合せ（未合成仮説）を取っておき、その後の推論で必要となった場合、そこから取り出すことになる。

以上の方法を利用して、最適解を高速に得る述語論理知識ベース対応の仮説推論システムを開発し、 KICK-HOPE II と名付けた。 5節に KICK-HOPE II のデータ構造、 システム構成、推論のアルゴリズムについて詳しく述べる。

5. KICK-HOPE II のアルゴリズム

知識ベース中の仮説は一般にルール型であってもよいが、 KICK-HOPE[4]や命題ホーン節の高速仮説推論法[2),3)]で採用されているように、 前処理として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延データベース (EDB)）に置き換える。たとえば、一般にルール型の仮説 p(X):-q(X) は完全知識 p(X):-q(X),h(X) と仮説 h(X) に置き換えることができる。

今、仮説はすべて単節であると考え、4.1項に基づきそれぞれの仮説の1以上の整数の重みを付ける。また、矛盾を表す知識は仮説間で定義され、あらかじめその仮説と組み合わせると矛盾を引き起こすような仮説の組合せの最小の集合 (minimal set) を計算しておく。矛盾が完全な知識、不完全な知識両方にまたがって与えられる場合は、その知識をゴールとしてサポートする最小の仮説の組合せを求め、それらが矛盾を表す知識となる。この計算自体は矛盾の可能性がなく、また、ゴールが与えられる以前に計算可能であるため、計算の効率性は大きな問題とはならない。

KICK-HOPE で取り扱うノードのデータ構造は図2で示されたように

＜ノード名, サポートする解仮説＞

となっていた。KICK-HOPE II においてはサポートする解仮説をすべて求める必要はないので、その代わりに最適解仮説とまだ計算していない仮説の組合せ、未合成仮説群とを置く。すなわち、次のようにする。

＜ノード名、最適解仮説、未合成仮説群＞

最適解仮説は 重み：仮説 の形をとる。未合成仮説群はリストの形をとり、その要素は （下界値、未合成仮説群） となっており、下界値によってソ

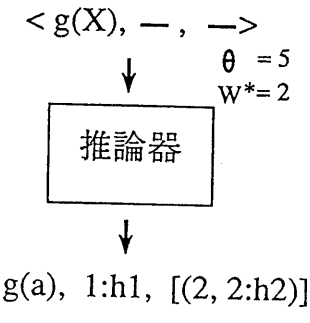


図6 KICK-HOPE II の推論例

ートする。

初期状態では、ノード名は変数を含んでいても構わず、最適解仮説及び未合成仮説群は未定である。ノードを重みの暫定値、しきい値と共に推論器にかけると、確定ノードが生成される（図６参照）。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。

KICK-HOPE II の確定ノードは次の４種に分類される。

1. 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. そのノードに対する最適解仮説が決まる。
   <ノード名, 最適解仮説, 未合成仮説群>
4. そのノードの解仮説は最適解になる可能性がない。
   <ノード名, non, 未合成仮説群>

4 は最適解となる仮説を計算している途中にその重みが暫定値を超えた場合に生じる。その場合でも、そのノードよりも上位のノードを計算する際に、未合成仮説群のデータを利用する場合があるので、未合成仮説は残しておく。

KICK-HOPE II の推論器の構成を図７に示す。あ

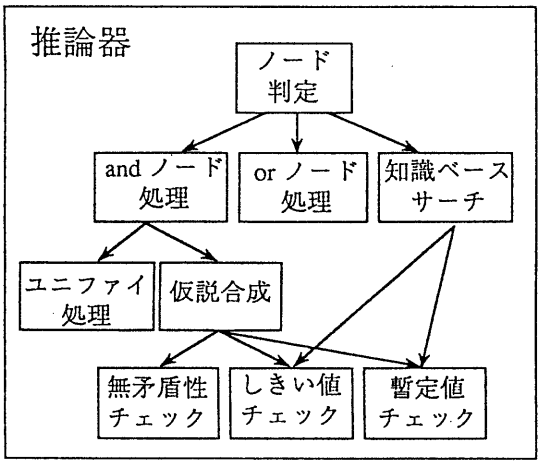


図７　KICK-HOPE II の推論器の構成

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。

ノートに対する重みの暫定値をω*、解仮説となる重みに対するしきい値をθとして、and ノード処理、 or ノード処理及び知識ベースサーチのアルゴリズムを以下に示す。

<<ノード N1 and N2 に対する and ノード処理>>

① 暫定値ω*、しきい値θとして、ノード N1 を解く。（１つまたは複数の確定ノード群 GN1 が生成される。）
② GN1 がない場合は終了。
③ GN1 から１つのノード FN1 を取り出し、残りのノード群を GN1' とする。ノード FN1 の変数値をノード N2 に伝播させ N2' とする。
④ 暫定値ω*、しきい値θとして、 ノード N2' を解く。（１つまたは複数の確定ノード群 GN2 が生成される。）
⑤ GN2 がない場合は GN1' を GN1 として②へ。
⑥ GN2 から１つのノード FN2 を取り出し、残りのノード群を GN2' とする。
⑦ ノード FN1 と ノード FN2 とで仮説合成 and 処理をする。 得られた解をノードの最適解仮説の重みによりソートする。 暫定値ω* を更新し、GN2' を GN2 として⑤へ。

<<ノード N1 or N2 に対する or ノード処理>>

① 暫定値ω*、しきい値θとして、ノード N1 を解く。（１つまたは複数の確定ノード群 GN1 が生成される。）暫定値ω*を更新する。
② 暫定値ω*、しきい値θとして、ノード N2 を解く。（１つまたは複数の確定ノード群 GN2 が生成される。）暫定値ω*を更新する。
③ GN1 と GN2 のうち同じノードに対しては、仮説合成 or 処理をし、 それぞれのノードの最適解仮説の重みによりソートする。

<<ノード N に対する知識ベースサーチ処理>>

ノード N 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。

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種に分類される。

・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ノード名に変えて返す。 すなわち、＜ボディ名,-,- ＞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 ＜ノード名,true,- ＞を返す。

・不完全知識（仮説）とユニファイされる場合：
　その仮説の重みをωとした場合、
　ω>θ→ false ノード
　　＜ノード名,false,- ＞
　ω>ω* →
　　＜ノード名,non,[(ω,ω：仮説)]＞
　それ以外 → ＜ノード名,ω:仮説,[]＞
　を返す。

・ユニファイされる知識がない場合：
　false ノード ＜ノード名,false,- ＞を返す。

仮説合成 and、仮説合成 or は ４.２項で述べた方法により、仮説の重みが小さくなる可能性の高い順（すなわち、下界値が小さい順）に合成する。 具体的な仮説合成処理は下記の通りである（図８参照）。

--- 仮説合成 and 処理　A & B

B より最小の重みの仮説 B_1を取り出し*、残りを B_OTH とする。同様に A からも重みの小さい順に順次仮説を取り出し、 それらを $A_1, A_2, \ldots$ とする。基本的に仮説の合成順序は Aの要素とB1の組合せを順次計算して行くが、 途中で （A_iの重み）＞（B_OTH の下界値）となった場合は、AとB_OTH 中の仮説の組合せの重みの方が A_iとB_1の組合せの重みよりも小さくなる可能性があるので、Aの要素とB_1の組合せ計算は一時保留にする。 そして、 B_OTH より最小の重みの仮説 B_2を取り出し、Aの要素とB_2の組合せを計算して行く。 その際に B_1は Aの要素の何番目までを計算したかという情報、すなわち、i を憶えておく。

--- 仮説合成 or 処理　A or B

A と B の下界値の小さい方を A とする。A より最小の重みの仮説を取り出す*。 その計算の途中で B の下界値を超えた場合は、 B の中により小さい重みの仮説が存在する可能性があるので A の計算は一時保留し、B に対する計算を開始する。B の下界値の方が小さい場合は A と B を逆にして考える。

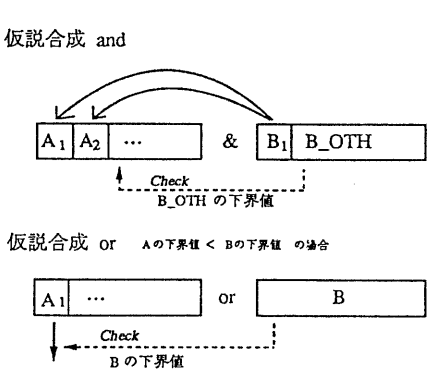


図８　仮説合成処理の概念図

いずれの場合も、仮説を合成させた時にその重みがしきい値θを超えた場合はその合成結果を削除する。 また、合成結果を得る毎に暫定値ω*は更新する。 残りの仮説の下界値がω*を超えた場合は仮説合成処理を終了する。まだ計算していない部分は下界値でソートして未合成仮説群とする。尚、 文中 * の処理は仮説合成 and と 仮説合成 or の組合せにより計算できる。

以上の推論法により、KICK-HOPE II は以前に開発した KICK-HOPE に備えられたゴールを証明するのに必要なノードのみを一度だけ推論するといった特長を持つと共に、仮説合成においては極力少ない合成回数で最適解仮説を得ている。

６．例題による推論速度の評価

例題を用いて、KICK-HOPE II の推論速度の評価を行う。例題としては、論理回路の故障診断の例を用いる。我々が論理回路の故障診断を仮説推論に応用する場合、 各々のゲートの状態 （正常、stack-on、stack-off）を仮説としている。 回路の規模が大きくなるとゲート数が増えるため、仮

説の数が増大し、どのゲートが故障しているかを判定するための組合せ計算が膨大になる。

このような例では、仮説に重みを付けることにより、KICK-HOPE II の推論法を有効に利用することができる。即ち、

ゲートが正常・・・ OK → 重み＝１

ゲートが異常・・・ ON, OFF → 重み＝N

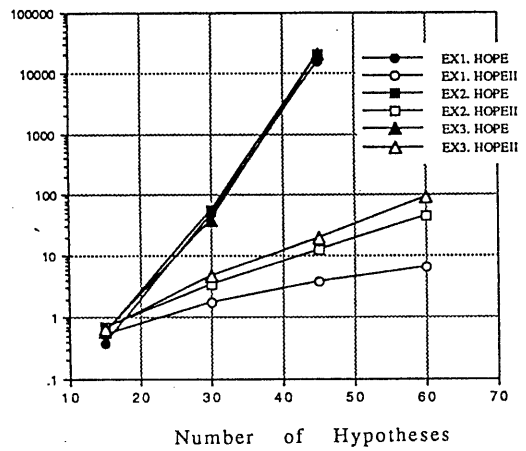
（Nは比較的大きい数）

とすることにより、 OK の状態を優先させて推論を進めることができる。一般の場合でも、故障箇所は１～２カ所であることが多く、すべての場合の組合せを計算するのは実用的ではない。尚、ここでのNは大きいほどより OK の状態が優先され、本例題ではNはゲート数程度にしてある。

また、KICK-HOPE II では、その他に仮説の重みの最大値としてしきい値を利用することができる。本例題では、故障状態のゲート数は３以下になるようにしきい値を設定している。

実験結果と解の個数を図９に示す。この実験結果により KICK-HOPE II では、解の個数が絞られたため推論時間が著しく短縮したことが解る。また、この実験で得られた最適解の結果は、EX1 は故障なし、EX2 は１カ所故障、EX3 は２カ所故障となっている。 KICK-HOPE II では、故障箇所が少ないものを優先して推論を進めるため、故障箇所が多いものほど推論時間が多くかかることを示している。しかし、一般には何カ所も同時に故障することはあり得ないので、最適解推論により比較的早期に結果を得ることができる。 尚、KICK-HOPE II では、実際の故障箇所が最適解と一致しない場合（例えば、故障が最適解では１カ所であるが実際には２カ所故障しているような場合）次善の解を得ることは容易にできる。

Inference Time
(CPU time [sec])



解の個数

仮説数		15	30	45	60
EX1	HOPE	19	453	11351	——
	HOPE II	1	1	1	1
EX2	HOPE	14	498	13006	——
	HOPE II	2	1	1	1
EX3	HOPE	14	356	13028	——
	HOPE II	1	1	1	1

図９　Adder の故障診断の実験結果

７．むすび

以上、述語ホーン知識を扱う高速仮説推論システム KICK-HOPE の最適解推論版 KICK-HOPE II について述べた。KICK-HOPE II は、仮説合成の組合せ数の増大という KICK-HOPE の問題点を克服している。しかし、仮説に対する矛盾制約知識が多いなど制約がきつい場合は、それほど仮説合成の回数は増えないので、KICK-HOPE の方が重みの判定などの処理を含まないため高速となる。 即ち、KICK-HOPE II は最適解を求めたい場合や仮説に対する制約が緩く、解の個数が大きくなると予想される場合に効果的となる。

また、KICK-HOPE II の推論時間は仮説の重みにより大きく影響を受ける。仮説の重みに差がある場合、一般に合成する仮説の数が減るので高速化が期待できる。特に矛盾により合成された仮説の組が棄却されることが多くなく、中間段階で得られた仮説の組合せが最終的な解の一部となる可能性が高い場合、大幅な効率化が期待できる。（６節の例題はこの場合である。）中間段階で得られた仮説の組合せがすべて後に矛盾となった場合は、次の候補を未合成仮説群の中から捜すことになり、最悪の場合は全解探索と同じすべての組合せを計算することになる。また、仮説の重みに差が少ない場合は、ほとんど全解探索をする場合と同じになり、KICK-HOPE II の優位性は現れにくい。

本システムと類似な考え方として、 Ng と Mooney による AAA/H Algorithm[12]がある。この方法では、仮説の重みを利用して解を絞る方法として４．２ 項で述べたビーム探索法により解の個数を制限する方法を利用している。しかし、この方法は４．２ 項で述べたようにビーム幅の設定が難しく、特に仮説推論では中間段階で得られた解が後の推論においてすべて矛盾する可能性も高いため、仮説推論には不向きであると思われる。

尚、仮説推論を始めとする非単調推論の理論的な計算複雑度は NP-完全、あるいは NP-困難と証明されており[13]、 本システムも通常の探索の枠組みを超えていないため、指数オーダの壁を超えることはできない。指数オーダの推論時間の壁を超えるため有効と思われる方法として、最適解の条件を緩和して準最適解を高速に得る方法[14]、過去の経験を利用した学習[15]や類推[16]の機構を利用する方法、 知識ベースのコンパイル法[17]などが考案されている。

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：推論パスネットワークによる高速仮説推論システム、人工知能学会誌、Vol.6, No.4, pp.501-509 (1991.7)
3) 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)
4) A.Kondo,T.Makino and M.Ishizuka: An Efficient Hypothetical Reasoning System for Predicate-logic Knowledge-base, Proc. Int'l Conf. on Tools for Artificial Intelligence (ICTAI'91), pp.360-367 (1991.11)
5) 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)
6) 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)
7) 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)
8) 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)
9) de Kleer.J.: An Assumption-based TMS, Artificial Intelligence, 28, pp.127-162 (1986)
10) 人工知能学会（編）：人工知能ハンドブック、オーム社、(1990)
11) 今野、鈴木：整数計画法と組合せ最適化、日科技連 (1982)
12) H.T.Ng and R.J.Mooney : An Efficient First-Order Horn-Clause Abduction System Based on the ATMS, AAAI-91, pp.494-499 (1991)
13) H.A.Kautz and B.Selman : Hard Problems for Simple Default Logics, Proc, KR'89, Toront, pp.189-197 (1989)
14) 岡本、石塚：0-1 整数計画法を用いて最適解を得る仮説推論法、情処学会人工知能研資料、81-7 (1992.3)
15) 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、知識のリフォメーション・シンポジウム論文集、情処学会 (1991)
16) 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、人工知能学会誌、Vol.7, No.1, pp.77-86 (1992)
17) 鶴田、石塚：発想的仮説生成のための述語論理知識ベースのコンパイル法、人工知能学会誌、Vol.6, No.1, pp.117-123 (1992)