論　文

推論の失敗を考慮した仮説推論システム

山ノ口　崇[†]　参沢　匡将[††]　木村　春彦[†]　小越　康宏[†††]
広瀬　貞樹[††††]

A Hypothetical Reasoning System Which Considers Failures in Reasonings

Takashi YAMANOKUCHI[†], Tadanobu MISAWA[††], Haruhiko KIMURA[†],
Yasuhiro OGOSHI[†††], and Sadaki HIROSE[††††]

あらまし　仮説推論は不完全な知識を扱うことができるので，設計や診断等の問題解決に有効であるが，仮説推論を含む非単調推論はNP-完全であることが知られており，低い推論速度が問題になっている．本研究では，推論木のANDノードの解を推論する際に，左の子と右の子のいずれを先に推論した方が効率がよいかを考察した．従来の1階述語論理を対象とした全数解を求める仮説推論システムでは，常に左の子の解を先に推論していたが，左の子の解が空ではなく，右の子の解が空のとき，左の子の解を求めた手間が無駄になる．このようなとき，右の子の解を先に求めていたら無駄は省けるが，左の子と右の子のいずれを先に推論したらよいかをどのように判別するかが問題となる．本論文では，この判別を容易に可能とする判別ラベルを導入し，これを用いて推論の失敗による無駄を排除した仮説推論システムを提案する．

キーワード　推論の失敗，空解の判定表，仮説推論，無矛盾性チェック

1. まえがき

仮説推論[1],[2]は，真か偽か分からない事柄をとりあえず真と考えて（仮説を立てて）推論を進め，矛盾なく観測事象を導くことができたら立てた仮説は正しかったと考える推論法である．診断[3]，設計[4]，プランニング[5]，自然言語理解[6]，画像認識[7]，スケジューリング[8]，文献検索[9],[10]など様々な分野に応用され，重要な推論の枠組みとなっているが，仮説推論の計算複雑度はNP-完全である[11]ため，効率的な推論は困難であり，実用化に向けて高速化が重要な課題となっている[12]．

仮説推論には，解を一つだけ求める場合（[13]～[17]など）と，すべての解を求める場合（[18]～[26]など）がある．前者は更に厳密解（最適解）を求める場合（[13],[15],[16]など）と近似解（準最適解）を求める場合（[14],[17]など）に分けられる．後者は，医師が処方せんを出すときのように，考えられるケースをできるだけ多く出した方が適切な処置ができる場合や，解に妥当なコストが付けられない場合に対象となる．また，知識ベースには，命題論理を対象にしたものと，1階述語論理を対象にしたものがある．1階述語論理は変数を用いることができるので命題論理よりも表現力が豊かであり，再帰表現も可能である．本論文では，1階述語論理を対象としたすべての解（全数解）を求める仮説推論を対象とする．

従来，1階述語論理を対象とした全数解を求める仮説推論システムでは，推論木のANDノードの解を推論する際に，常に左の子の解を先に推論していたが，左の子の解が空ではなく，右の子の解が空（推論の失敗）のとき，左の子の解を求めた手間が無駄になる．このようなとき，右の子の解を先に求めていたら無駄は省けるが，左の子と右の子のいずれを先に推論した方がよいかをどのように判別するかが問題となる．本論文では，判別ラベルを導入して空解となるノードを

[†] 金沢大学大学院自然科学研究科，金沢市
Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa-shi, 920-8667 Japan
[††] 東京理科大学経営学部経営学科，久喜市
School of Management, Tokyo University of Science, Kuki-shi, 346-8512 Japan
[†††] 福井大学工学部知能システム工学科，福井市
Faculty of Engineering, Fukui University, Fukui-shi, 910-8507 Japan
[††††] 富山大学工学部知能情報工学科，富山市
Faculty of Engineering, Toyama University, Toyama-shi, 930-8555 Japan

求め，そのノードへの弧を枝刈りすることにより仮説推論の高速化を図る．また，空解には仮説が存在しないために起こる場合と，矛盾によって起こる場合がある．これまで，仮説が存在しないときの空解による枝刈りについては，空解をボトムアップに伝搬させて枝刈りする手法が一般的に知られている．本論文では，仮説が存在しない場合だけでなく，矛盾による空解も含め，更に有向閉路の枝刈りまでを対象として，推論の失敗による無駄を排除する仮説推論システムを提案する．なお，判別ラベル付けの手間が小さいため，推論の失敗がなかったときのオーバヘッドがほとんど問題にならないことを実験により示す．

以下に，**2.** で従来の仮説推論システムを概説し，**3.** で提案手法に最も近い効率化に関する従来手法を説明し，**4.** で提案手法を示す．そして **5.** で従来手法と提案手法の比較実験を行い，**6.** で提案手法を評価する．

2. 従来の仮説推論システム

本論文では，1 階述語論理を対象とする従来の仮説推論システムとして KICK–HOPE を扱う．KICK–HOPE は演繹データベースの手法である QSQR 法を応用した高速仮説推論システム [21] である．このシステムは，関数を含まない述語ホーン節知識から，推論木と呼ばれるゴール指向のネットワーク（AND/OR 木）を生成する．ただし，仮説は変数を含まないファクト型の知識に制限されている．推論木のノードは単一リテラルからなるノード（単ノード），AND 関係，OR 関係の複合ノードである AND ノード，OR ノードの 3 種に分類される．ノードの情報は次のようなデータ構造で表される．

$< node, solutions >$

ここで，

node　　：ノード

solutions：ノードの解（このノードを真とする支持仮説の組の集合）

である．

推論はゴールから始まり，未確定ノード，つまりノードの変数の束縛値，及び支持仮説の組の集合が定まっていないノードについてそれらを求めていく後向き推論と，すべての子ノードが確定ノードになった親ノードを確定ノードにして仮説合成をする前向き推論を並列に行う．つまり，以下の (1)～(3) の処理を，ゴールを初期ノードとして，必要なすべてのノードに対して行う．

（1） 現在の推論対象のノードに対して未確定の子ノードがあれば，その子ノードを確定ノードにすべく推論を行う（後向き推論）．

（2） それぞれの子ノードに対し AND ノードがあれば，変数の束縛条件を満たすように値を伝搬しながら処理を繰り返す．

（3） 子ノードがすべて確定ノードになったら，親ノードを確定ノードにすべく，それぞれの子ノードの支持仮説の組の集合を合成して親ノードの支持仮説の組の集合とする（前向き推論）．

このように KICK–HOPE は，バックトラックと再計算を回避したゴール指向の前向き推論でゴールの証明に必要なノードのみを一度計算することにより，効率的な解仮説の生成を行うことができる．

[例 1]　次のような背景知識，仮説知識，ゴールからなる例を用いて KICK–HOPE の推論方式を説明する．まず，背景知識から図 1 のような AND/OR グラフを生成する．KICK–HOPE はこの AND/OR グラフを深さ優先探索で仮説に対応する葉を探していき，仮説合成により解を求める．また，有向閉路がある場合には展開をしながら仮説合成を行う．この操作を繰り返すことにより図 2 のような推論木が生成される．ここで，二重丸（◎）の葉は対応する仮説が存在するノードであり，一重丸（○）の葉は対応する仮説が存在しないノードである．

○背景知識

$f(X,0) \leftarrow f(X,1).$
$f(X,0) \leftarrow f(X,2).$
$f(X,1) \leftarrow f(X,3), f(X,4).$
$f(X,3) \leftarrow f(X,7), f(X,8).$
$f(X,4) \leftarrow h(X,7), h(X,8).$
$f(X,7) \leftarrow f(X,11).$
$f(X,7) \leftarrow h(X,4).$
$f(X,8) \leftarrow h(X,5).$
$f(X,8) \leftarrow h(X,6).$
$f(X,11) \leftarrow f(X,12), f(X,13).$
$f(X,12) \leftarrow h(X,0).$
$f(X,12) \leftarrow h(X,1).$
$f(X,13) \leftarrow h(X,2).$
$f(X,13) \leftarrow h(X,3).$
$f(X,2) \leftarrow f(X,5).$
$f(X,2) \leftarrow f(X,6).$

$f(X,5) \leftarrow h(X,9), h(X,10).$
$f(X,6) \leftarrow f(X,9), f(X,10).$
$f(X,9) \leftarrow h(X,11).$
$f(X,9) \leftarrow h(X,12).$
$f(X,10) \leftarrow f(X+1,6), h(X,13).$
$inconsistent \leftarrow h(X,0), h(X,6).$
(矛盾知識 1)
$inconsistent \leftarrow h(X,0), h(X,3).$
(矛盾知識 2)
$inconsistent \leftarrow h(X,12), h(Y,13), X \neq Y.$
(矛盾知識 3)

○仮説知識
$h(0,0).$
$h(0,3).$
$h(0,4).$
$h(0,5).$
$h(0,6).$
$h(0,7).$
$h(0,8).$
$h(0,9).$
$h(0,10).$
$h(0,12).$
$h(0,13).$
$h(1,12).$
$h(1,13).$
.
.
.
$h(n,12).$
$h(n,13).$

○ゴール
$f(0,0).$

以下に具体的に示す．まず，ゴール $f(0,0)$ から変数 X の値が 0 に確定する．ノードの制御の順番は，左側から探索する深さ優先探索に従う．つまり，図 2 の $f(0,0)$，$f(0,1)$，$f(0,3)$，$f(0,7)$，$f(0,11)$，$f(0,12)$，$h(0,0)$ の順に制御が移り，$h(0,0)$ で初めて確定ノードとなる．$f(0,12)$ に戻り，まだ調べていない $h(0,1)$ をチェックするが，仮説として存在しないので，$f(0,12)$ の解は $h(0,0)$ に確定する．$f(0,11)$ に

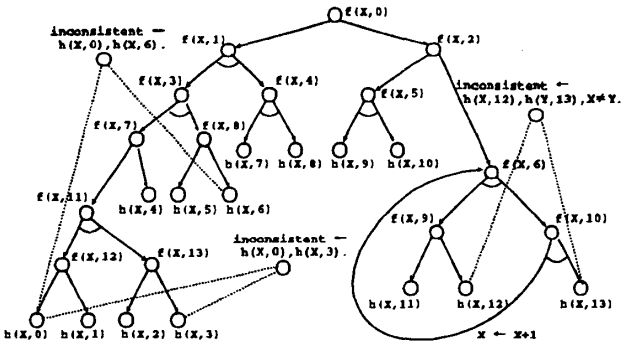


図 1　有向閉路を有する AND/OR グラフ
Fig. 1　AND/OR graph with a circuit.

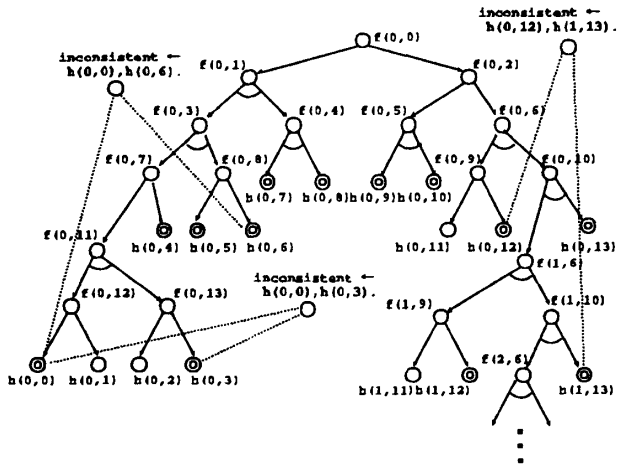


図 2　図 1 の推論木
Fig. 2　Reasoning tree for AND/OR graph in Fig.1.

戻り，$f(0,13)$，$h(0,2)$，$f(0,13)$，$h(0,3)$，$f(0,13)$，$f(0,11)$ と制御が移っていくが，仮説 $h(0,2)$ が存在しないので，$f(0,13)$ の解は $h(0,3)$ となり，$f(0,11)$ の仮説合成の結果は $h(0,0)$，$h(0,3)$ となるが，$f(0,11)$ が AND ノードであるので各矛盾知識に対して無矛盾チェックを行うと，矛盾知識 2 から $f(0,11)$ の解は空となる．以下同様に，$f(0,7)$，$h(0,4)$，$f(0,7)$，$f(0,3)$，$f(0,8)$，$h(0,5)$，$f(0,8)$，$h(0,6)$，$f(0,8)$，$f(0,3)\cdots$ の順に制御が移り，ノードの解が確定していく．

次に有向閉路に対しては，図 2 の右側で示されているように，上記の深さ優先探索に従って未確定ノードを確定ノードにする際に必要となる展開を繰り返していく．　□

3. 関連する従来研究

本研究に関連する従来研究は，(1) 仮説推論の高速化に関するもの [19], [22], [23], [25], [26] などと，(2) それ以外の高速化に関するもの [27], [28] などに分けられる．更に，(1) は，(1a) 推論実行前（仮説合成をす

る前）に行われるもの[22],[23]などと，(1b) 推論実行中に行われるもの[19],[25],[26]などに分けられる．本提案手法は，(1a) に分類される．

ここで，[22],[23],[25],[26]は，本提案手法と同様に，1階述語論理を対象とした全数解を求める仮説推論システムの高速化を対象としているのに対し，[19]は命題論理を対象とした全数解を求める仮説推論システムの高速化を対象としている．

関数を含まない1階述語論理表現はエルブラン領域で命題論理へ展開が可能であるので，変換効率の良い[28]の手法などを用いて，述語論理の高速化を命題論理の高速化手法で実現させることが可能である．しかしながら，このようなアプローチは，節数が膨大になってしまうため実用的には問題がある．

したがって，本提案手法と特に密接に関連する従来研究は，[22],[23],[25],[26]である．

[22]は，推論前に推論時間と所要メモリ量の両方の観点から有効となるアトムに関して，それをゴールとし，その他すべてのアトムを要素仮説とみなして制約条件を満たす範囲内での全解探索仮説推論を行う部分コンパイルにより実行時段階の矛盾処理や包摂処理を事前に行っている．[23]は，推論前に述語論理で表現された知識ベースを命題論理に抽象化することにより，実際の推論において計算すべき探索空間を近似的に求め，矛盾処理の不必要な探索空間の絞り込みを行う手法である．また，[25]は，矛盾処理時に用いる探索木から得られる情報を記憶しておき，後の矛盾処理時にその情報を用いて処理の軽減を図るものである．更に，[26]は，本提案と同じくKICK–HOPE[21]を対象としているが，推論実行中にチェックベクトルを導入して，矛盾処理，包摂処理の高速化を行う手法を提案している．これは，冗長な計算を削減するために，仮説集合に含まれる仮説をチェックベクトルで表し，このチェックベクトルを調べることで無駄な処理を削減するように改良したものである．このように上記に示したいずれの高速化手法も，仮説推論における矛盾処理，包摂処理を高速化する，というアプローチをとっている．

一方，先に示した (2) に分類される[27],[28]などの高速化手法は，推論に用いない知識を早い段階で除去するというアプローチをとっており，[22],[23],[25],[26]とは異なった考え方に基づき高速化を図っている．ただし，仮説推論の高速化を対象としたものではないので，このまま利用することはできない．本論文では，このアプローチを仮説推論に適用するための一手法を提案する．具体的には，判別ラベルを導入して空解となるノードを求め，そのノードへの弧を枝刈りすることにより仮説推論の高速化を図る．これにより，これまで十分な高速化が達成されていなかった述語論理の仮説推論に更なる高速化が期待できる．

また，前者 (1a) に分類される二つの手法[22],[23]は，本提案手法である「空解の伝搬による枝刈り」を行った後に実行できる．更に，後者 (1b) に分類される[25],[26]の手法も，本提案手法の適用した後に実行することができる．したがって，本提案手法とこれらの従来手法を組み合わせて用いればいっそうの効果が期待できる．**4.4**では，本提案手法がこれらの従来手法の効果を妨げないことを示す．また**5.**では，本提案手法を[26]の手法と組み合わせて用いたときにいっそうの効果が出ることを比較実験により示す．

4. 提案手法

4.1 諸定義

有向閉路上のノードをサーキットノードと呼び，サーキットノードの子孫をバリアブルノードと呼ぶ．また，本論文で扱う矛盾知識は2項矛盾に限定するが，2項矛盾 $inconsistent \leftarrow h1(x), h2(y)$ $[x = y \quad \text{or} \quad x \neq y]$ の $h1(x)$, $h2(y)$ に対応する二つの葉の共通の先祖で，ANDノードとなる高さ最小のノードを準矛盾ノードと呼ぶ．ただし，準矛盾ノードから $h1(x)$ までの有向道と，準矛盾ノードから $h2(x)$ までの有向道に共通して含まれる弧があってはならない．

4.2 判別ラベル

生成されたAND/ORグラフのノードを対象に，仮説知識と矛盾知識を用いて以下のように葉から順次，先祖に向かって判別ラベルを付加する．ここで，判別ラベルとは次のとおりである．

- 0：支持する仮説がない（解仮説なし）．
- 1：すべての解仮説に2項矛盾の1項が共通に含まれている．
- 2：サーキットノードまたはバリアブルノードであり，閉路を展開しないと空解になるかどうかは分からない．
- 3：その他．

[判別ラベルの付加手続き]

（i） 葉に判別ラベルを付加する．

（ii） 二つの子に判別ラベルが付加されているノー

ド D に判別ラベルを付加する．ただし，D が AND ノードのときは表 1 を用いて，また OR ノードのときは表 2 を用いて判別ラベルを付加する．ここで，表 1，表 2 の LDD，RDD はそれぞれ D の左直子孫（左の子），右直子孫（右の子）を表す． □

判別ラベルは，そのラベルが付加されたノードの解仮説の状況を表すものであり，空解により枝刈りが可能であるとか，すべての解仮説に 2 項矛盾の 1 項が共通に含まれていて，先祖の準矛盾ノードで矛盾知識により空解になる可能性があるとか，空解になるかどうかは AND/OR グラフを展開しないと分からないとか，空解にはならないといったことを表すものである．具体的には，0 が付加されたノードは解が空（空解）であり，このノードに入ってくるすべての弧を枝刈りすることができる．1 が付加された AND ノードは空解になる可能性があり，その AND ノードが準矛盾ノードで，ある 2 項矛盾の各項に対応してその二つの子がともに 1 が付加されているとき空解となり，準矛盾ノードへ入ってくる弧をすべて枝刈りすることができる．この空解の判別は **4.3** で示すアルゴリズムの操作 4 により行われ，その際，判別ラベルは付け直される．2 が付加されたノードは自分自身が含まれている閉路や，先祖の閉路を展開しないと空解になるかどうかは分からない．3 が付加されたノードは空解になることはないので枝刈りすることはできない．

表 1　AND ノードの判別ラベル
Table 1　Distinction label for AND node.

LDD	*RDD*	AND
0	0	0
0	1	0
0	2	0
0	3	0
1	0	0
1	1	1
1	2	1
1	3	1
2	0	0
2	1	1
2	2	/
2	3	/
3	0	0
3	1	1
3	2	/
3	3	3

表 2　OR ノードの判別ラベル
Table 2　Distinction label for OR node.

LDD	*RDD*	OR
0	0	0
0	1	1
0	2	/
0	3	3
1	0	1
1	1	1†
1	2	/
1	3	3
2	0	/
2	1	/
2	2	/
2	3	3
3	0	3
3	1	3
3	2	3
3	3	3

†：左右の子の矛盾の項が異なるときは 3 である．

表 1，表 2 について説明する．まず表 1 であるが，これは対象とするノード X が AND ノードの場合である．少なくとも一方の子に 0 が付加されていれば，X は空解となる．それゆえ，X には 0 が付加される．また，少なくとも一方の子に 1 が付加されており，各子とも空解とならないとき，X のすべての解仮説に 2 項矛盾の 1 項が共通に含まれる．それゆえ，X には 1 が付加される．更に，各子とも 3 が付加されていれば，X が空解となることはないので 3 が付加される．なお，/は閉路を展開しないと分からないものであり，後で決定するものである．しかし，左右の子にそれぞれ 1，2，若しくは 2，1 が付加されるときは，X に付加される可能性のある判別ラベルは 0 か 1 であるので，1 とみなす．これは，必ず空解になるかどうかを調べたいからである．

次に表 2 であるが，これは対象とするノード X が OR ノードの場合である．論理和の定義から一方の子に 0 が付加されているときは，残りの子に付加されている判別ラベルが付加される．ただし，残りの子に 2 が付加されているときは，0，1，3 の可能性がある．なお，一方の子に 2 が付加されていても，残りの子に 3 が付加されているときは，いずれにせよ 3 となる．一方の子に 2 が付加されている他の組合せ（残りの子に 1 または 2 が付加される場合）では，0，1，3 の可能性がある．左右の子に 1 が付加されているときは，各子においてすべての解仮説に共通に含まれている 2 項矛盾の 1 項が左右の子とも同一であれば，1 が付加され，異なれば 3 が付加される．この左右の子の矛盾の項が異なるときに 3 となるのは，ノードに付与する情報が判別ラベルだけであり，各矛盾知識ごとに処理しているからである．そして，一方の子に 3 が付加さ

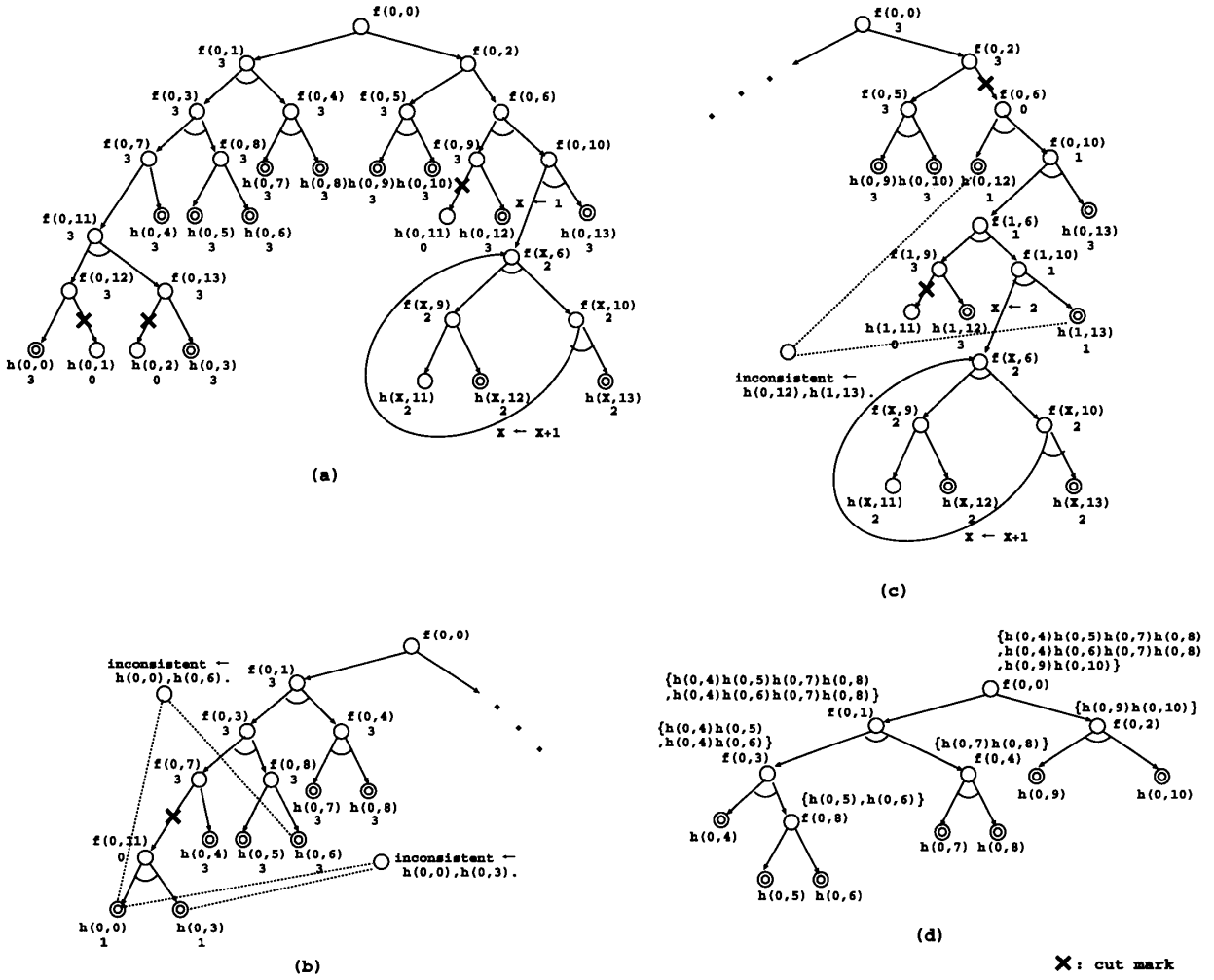


図 3 例 1 の知識ベースに対する枝刈りされた AND/OR グラフ
Fig. 3 Reformed AND/OR graph for the knowledge base of Example 1.

れていれば，残りの子にどのような判別ラベルが付加されていても 3 となる．

4.3 アルゴリズム

(操作 1) 知識ベースから等価な AND/OR グラフを作成する．

(操作 2) ゴールに対応するノード G をルートノードとする部分グラフ（ノード G 及び G のすべての子孫から構成されるグラフ）を抜き出し，以降の操作をこのグラフに対して行う．なお，ゴールのパラメータ値を対応する変数に代入する．ただし，有向閉路がある場合には，まず，有向閉路を 1 回だけ展開し，残った有向閉路のサーキットノードやバリアブルノードの述語のラベルの変数にはパラメータ値を代入しない．

(操作 3) 矛盾知識は使わず，仮説知識のみを用いて判別ラベルを付加する．また，判別ラベル 0 が付加されたノードへの弧を枝刈りする．

(操作 4) 2 項矛盾の項に対応する二つの葉がいずれもバリアブルノードになっていない各矛盾知識に対し，次の操作を繰り返す．ただし，ノードに付与する情報は判別ラベルだけであり，各矛盾知識ごとに処理をする．

(1) 2 項矛盾の二つの項に対応する葉から準矛盾ノードの子までのノードに判別ラベルを付加する．ただし，既に付加されていた判別ラベルは改めて付け直す．

(2) 準矛盾ノードの二つの子の判別ラベルがともに 1 のとき，準矛盾ノードの判別ラベルを 0 に変更する．また，この変更により，準矛盾ノードの先祖に対し，判別ラベルを付け直す．この作業は 0 以外の判別ラベルが付加されるまで続ける．更に，判別ラベル 0 が付加されたノードへの弧を枝刈りする．

(操作 5) 有向閉路を更に 1 回展開し，生成された推

論木部分に対し，かかわってくる仮説知識と矛盾知識を用いて，操作3，操作4と同様の操作により判別ラベルを付加し，判別ラベル0が付加されたノードへの弧を枝刈りする．ただし，生成された推論木部分の変数にはパラメータ値を代入する．

(操作6)　有向閉路が残っていれば操作5へ飛び，残っていなければ最終的に残った推論木に対して仮説合成を行ってルートノードの解を求める．ただし，無矛盾性チェックを必要とするノードは推論木内の準矛盾ノードだけである．　□

[例2]　提案手法を例1の知識ベースを用いて具体的に説明する．まず，操作1によりAND/ORグラフを生成し，操作2でゴールをルートノードとする部分グラフを作成する．操作3の結果を図3(a)に示す．この段階で仮説知識だけを用いて枝刈りが行われる．もし，仮説 $h(0,7)$ か $h(0,8)$ のいずれかが存在しなければ，$f(0,0)$ と $f(0,1)$ を結ぶ弧が枝刈りされていた．従来方式では左の子，右の子の順に処理をするため，$f(0,3)$ の解を求めてから $f(0,4)$ の空解に気がつくため大きな無駄をするところである．操作4の結果を図3(b)に示す．これは有向閉路にかかわってこない矛盾知識による枝刈りである．操作5の結果を図3(c)に示す．有向閉路を計2回展開しただけで有向閉路が枝刈りされている．操作6の結果を図3(d)に示す．もとのAND/ORグラフの部分グラフに絞り込まれ，仮説合成によって解が求められている．　□

4.4 従来手法と提案手法の独立性

提案手法は，1階述語論理の知識ベースから等価なAND/ORグラフを作成し，空解となるノードを求めて，そのノードへの弧を枝刈りすることにより仮説推論の高速化を図っている．つまり，枝刈り後のAND/ORグラフはもとのAND/ORグラフの部分グラフとなる．また，枝刈り後のAND/ORグラフから等価な知識ベースを生成すれば，本提案手法と同様に1階述語論理の知識ベースを扱う（競合する）従来手法[22],[23],[25],[26]が実行可能となる．すなわち，本提案手法はこれらの従来手法の前に実行可能であり，かつ，これらの従来手法の効果を妨げるものではない．なぜならば，本提案手法では，もとのAND/ORグラフの部分グラフを生成し，新たに追加されるノードや弧がないからである．

ただし，枝刈りによる計算量の軽減と，組み合わせて実行する従来手法の計算量の軽減の和が，単独で従来手法を実行するときの軽減よりも小さい場合もある．例えば，枝刈りができないときや，できても葉に近い弧の枝刈りのため，軽減される計算量が小さいときには，推論時間が悪化してしまう場合がある．最悪の場合，提案手法のオーバヘッド分だけ大きくなる．

5. 実　　験

提案手法の有効性を検証するためにKICK–HOPEとの比較実験を行った．最初に例1の知識ベースを用いた場合の実験結果を表3に示す．表3から，有向閉路のループ回数 n が大きいほど提案手法が有効であることが分かる．これは，提案手法では有向閉路部分が枝刈りされているため，有向閉路のループ回数に依存せずに一定の推論時間で解けるのに対し，KICK–HOPEでは最終的に必要のない有向閉路部分の探索や仮説合成を行うため，推論時間が指数関数的に増加するからである．また，展開しても枝刈りができない場合を試すために，矛盾知識3を削除した場合を実験した．その結果を表4に示す．提案手法の方が0.08～0.09秒大きくなっているが，これはほとんどラベル付けの準備に要する手間であり，主にラベル付け用のネットワーク作成の手間である．ラベル付け自体はほどんど時間がかかっていない．KICK–HOPEではネットワーク

表3　KICK–HOPEと提案手法の比較（例1の知識ベース）

Table 3　Comparison between KICK–HOPE and the proposed method (knowledge of Example 1).

n	Proposed [s]	KICK–HOPE [s]
1	0.20	0.23
5	0.20	0.25
10	0.20	0.28
20	0.20	0.45
30	0.20	1.06
40	0.20	2.43
50	0.20	5.25

表4　KICK–HOPEと提案手法の比較（例1の知識ベースから矛盾知識3を削除したとき）

Table 4　Comparison between KICK–HOPE and the proposed method (knowledge of Example 1 from which inconsistency 3 was removed).

n	Proposed [s]	KICK–HOPE [s]
1	0.20	0.11
5	0.23	0.15
10	0.25	0.17
20	0.29	0.20
30	0.50	0.42
40	0.98	0.90
50	2.01	1.93

を作りながら推論を行うが，提案手法ではラベル付けのためにネットワークをあらかじめ作成し，その後にKICK–HOPEを使用する．ただし，ラベル付け用のネットワークは有向閉路を展開せずにラベル付けを行うため，反復回数が増えてもラベル付け自体がほとんど時間がかからないため手間はほとんど増えていないと考えられる．また，反復回数 n が1のときと，20のときだけが，提案手法とKICK–HOPEの手間の差が0.09秒となり，その他が0.08秒となったのはタイマの誤差によるものと思われる．

上述の実験から提案手法により有向閉路が枝刈りされた場合に効果が大きいことが分かった．有向閉路の枝刈りの効果は，枝刈りできるループの反復回数が大きいほど大きくなる．次に有向閉路を含まない場合の有効性を検証するために，有向閉路を含まない知識ベースをランダムに作成し，実験を行った．知識ベースの作成については，推論木の深さが8〜10となるようにし，各葉の深さも乱数を用いて変動させた．また，ノードをANDノードにするかORノードにするかも乱数を用いて決めた．更に，矛盾知識を乱数を用いて作成し，数は0〜10になるようにした．ここでは本提案手法の枝刈りの効果の差異が，ある程度明確に出ると思われる規模で，また，現実的にはいろいろな知識ベースが考えられるので，実験する知識ベースに対応する推論木のノードの種類に偏りが出ないようにするために，このようなパラメータ設定を行った．結果を表5に示す．提案手法の()内の数値は判別ラベルの付加手続きに要した時間である．HDTは仮説が存在しないことによって空解となるノードへの弧を枝刈りすることによる推論の削減時間であり，IDTは矛盾知識によって空解となるノードへの弧を枝刈りすることによる推論の削減時間である．表5から，KB9とKB10以外の知識ベースにおいて提案手法の効果が大きく，判別ラベルの付加手続きにほとんど時間を要していないことが分かる．また，解仮説数が多くなるほど差が顕著である．これは，解仮説が多いほど各ノードにおける仮説合成が多くなるため，枝刈りにより必要のない仮説合成を省略できたことによる効果であると考えられる．また，知識ベースKB7では特に効果が大きい．この知識ベースでは最終的に解仮説数が0であるため，KICK–HOPEでは左の子の解を求めてから右の子が空解であることに気がついたのに対し，提案手法では判別ラベルにより直に空解が分かったことによる差であると考えられる．しかし，知識ベースKB8のようにあまり効果が得られない場合もある．これは，提案手法によって行われる枝刈りが葉付近であるため，省略できる仮説合成が少なかったためであると考えられる．KB9とKB10は枝刈りが存在しない場合であり，ラベル付けのオーバヘッド分だけが加算されている．表5には，10例挙げたが，実験では22例について調べた．その結果，類似しているものが多かったのでそれらを削除した．なお，表5の中で矛盾知識がなかったものはKB2とKB5である．

以上より，本提案手法は有向閉路を枝刈りできる場合は効果があり，特にループ回数が多い場合には顕著である．また，有向閉路を含まない場合，枝刈りできるノードが葉から遠ざかるほど効果が大きいといえる．しかし，枝刈りが葉付近で生じた場合や，枝刈りできない場合には効果が出ない．これは枝刈りによるコストの軽減よりも判別ラベルを付加する手間の方が大きくなる場合であるが，本実験の範囲内では判別ラベル付けのオーバヘッドは推論時間に比べ，極めて小さい．

次に，表5と同じ知識ベースを用いて，文献[26]の手法と組み合わせて用いた場合の効果を調べた．提案

表5　KICK–HOPEと提案手法の比較（ランダム知識ベース）
Table 5　Comparison between KICK–HOPE and the proposed method (random knowledge base).

	Proposed [s]	HDT [s]	IDT [s]	KICK–HOPE [s]	solutions
KB1	0.290(0.015)	0.00	0.40	0.640	2
KB2	0.420(0.035)	0.35	0.00	0.740	40
KB3	8.350(0.300)	4.64	10.52	23.460	760
KB4	74.580(0.120)	19.46	83.28	177.280	3186
KB5	771.950(0.250)	653.64	0.00	1425.530	9666
KB6	1053.050(0.330)	850.34	1808.84	3712.170	11360
KB7	2.940(0.250)	0.00	127.60	130.490	0
KB8	264.780(0.035)	23.63	0.00	288.360	8624
KB9	74.500(0.390)	0.00	0.00	74.450	4312
KB10	7892.820(0.220)	0.00	0.00	7892.760	35644

表 6 提案手法を文献 [26] と組み合わせた場合の効果
Table 6 The effect of combining our proposed method and the method in literature [26].

	Proposed+ [26] [s]	[26] [s]
KB1	0.590	0.720
KB2	0.680	0.800
KB3	4.620	6.950
KB4	10.600	17.020
KB5	38.620	70.150
KB6	44.880	105.780
KB7	3.110	20.140
KB8	29.520	31.980
KB9	16.550	16.110
KB10	207.500	207.240

手法の後に文献 [26] の手法を実行した場合と，単独で文献 [26] の手法を実行した場合の実行時間を比較した．その結果を表 6 に示す．KB1～KB8 はいずれも効果が現れているが，KB9 と KB10 は悪化してしまっている．これは，いずれも枝刈りができなかったケースであり，提案手法のオーバヘッド分だけ大きくなっている．

また，KB1，KB2，KB7 は仮説合成数が少ないために，文献 [26] の手法のオーバヘッドの影響が大きくなり，提案手法単独よりも，文献 [26] の手法を組み合わせた方が実行時間が大きくなった．

6. 評　　価

このアルゴリズムで枝刈りされ，最終的に残った推論木には推論の失敗がない．それは各ノードに判別ラベルを付加させることにより，仮説がなかったり，2 項矛盾により空解となるノードへ入る弧をあらかじめ枝刈りしているからである．また，この推論木の準矛盾ノードでは，必ず対応する 2 項矛盾が起こる．これは，準矛盾ノードから 2 項矛盾の各項に対応する葉までの有向道上のノードに判別ラベル 0 が付加されないからである．もし付加されるのであれば，そのノードへの弧は枝刈りされていなければならないので矛盾する．更に，準矛盾ノード以外では矛盾が起こらないので，準矛盾ノードで対応する 2 項矛盾の無矛盾性チェックだけを行えばよい．これに対し，KICK–HOPE では各 AND ノードで全 2 項矛盾に対し無矛盾性チェックをしなければならない．実験では，提案手法がほとんどのケースで勝っていたが，提案手法が悪化するケースもある．それは，矛盾知識がなく，空解となる内部ノードがない場合である．また，主な差は判別ラベルを付加する手間の分であるが，表 5 から分かるとおりその手間は極めて小さい．更に，表 6 から，提案手法を文献 [26] の手法と組み合わせることにより効果が出ることが分かった．

7. む　す　び

本論文では，空解となるノードを捜すための判別ラベルを導入し，空解となるノードへの弧を枝刈りすることにより無駄な推論を排除する手法を提案した．これにより推論の高速化が期待できる．

謝辞　査読者の先生方から有益な御教示，御助言を頂いた．ここに記して謝意を表する．

文　　献

[1] 國藤　進，“仮説推論，” 人工知能誌，vol.2, no.1, pp.22–29, 1987.
[2] D. Poole, R. Aleliunas, and R. Goebel, “Theorist: A logical reasoning system for defaults and diagnosis,” in The Knowledge Frontier : Essays in the Representation of Knowledge, Springer-Verlag, N.Y., 1987.
[3] R. Reiter, “A theory of diagnosis from first principles,” Artif. Intell., vol.13, pp.57–95, 1987.
[4] 牧野俊郎，石塚　満，“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，” 人工知能誌，vol.5, no.5, pp.640–648, 1990.
[5] K. Eshghi, “Abductive planning with event calculus,” Proc. 5th ICLP, pp.562–579, 1988.
[6] J. Hobbs, M. Stickel, D. Appelt, and P. Martin, “Interpretation as abductions,” Artif. Intell., vol.63, pp.69–142, 1993.
[7] 辻野広司，エドガー　ケルナー，桝谷知彦，“画像認識のためのマルチエージェントによる仮説推論，” 人工知能誌，vol.12, no.3, pp.440–447, 1997.
[8] 河原耕治，山本　剛，佐々木博司，久保川淳司，朝原春海，“仮説推論を用いた停電作業スケジューリングに関する一考察，” 電学論（B），vol.118, no.4, pp.460–466, 1998.
[9] 大澤幸夫，須川敦史，谷内田正彦，“ユーザーの変化する興味を理解し表現する文献検索支援システム Index Navigator，” 人工知能誌，vol.13, no.3, pp.461–469, 1998.
[10] 松村真宏，大澤幸夫，谷内田正彦，“AAS：文書の組み合わせによってユーザの興味を満足する検索システム—コストに基づく仮説推論の応用，” 人工知能誌，vol.14, no.6, pp.1177–1185, 1999.
[11] T. Eiter and G. Gottlob, “The complexity of logic-based abduction,” J. Association for Computing Machinary, vol.42, no.1–2, pp.3–42, 1995.
[12] 石塚　満，“仮説推論の計算量と高速化メカニズム，” 人工知能誌，vol.9, no.3, pp.342–349, 1994.
[13] E. Santos, Jr. and E.S. Santos, “Polynomial solvability of cost–based abduction,” Artif. Intell., vol.86, pp.157–170, 1996.
[14] 大澤幸夫，石塚　満，“多項式時間仮説推論を達成するネットワーク化バブル伝搬法の述語論理への拡張，” 人工知能誌，vol.10, no.5, pp.731–740, 1995.

[15] 加藤昇平，世木博久，伊藤英則，“PARCAR：コストに基づく並列仮説推論システム，” 情処学論，vol.41, no.3, pp.668–676, 2000.
[16] 加藤昇平，中村智典，伊藤英則，“ワークステーションクラスタを用いた並列仮説推論システム，” 信学論（D-I），vol.J84-D-I, no.9, pp.1412–1420, Sept. 2001.
[17] M. Ishizuka and Y. Matsuo, “SL method for computing a near-optimal solution using linear and non-linear programming in cost-based hypothetical reasoning,” Knowledge-Based System, Knowledge-Based Systems, vol.15, no.7, pp.369–376, 2002.
[18] J. de Kleer, “An assumption-based TMS,” Artif. Intell., vol.28, pp.127–162, 1986.
[19] 伊藤史郎，石塚　満，“推論パスネットワークによる高速仮説推論システム，” 人工知能誌，vol.6, no.4, pp.501–509, 1991.
[20] 牧野俊朗，石塚　満，“経験に基づく学習による仮説推論の高速化，” 人工知能誌，vol.8, no.3, pp.320–327, 1993.
[21] 近藤朗子，牧野俊朗，石塚　満，“述語論理知識を扱う高速仮説推論システム，” 人工知能誌，vol.8, no.6, pp.819–827, 1993.
[22] 堂前宣夫，石塚　満，“仮説推論高速化のための知識ベースリフォーメーション，” 人工知能誌，vol.9, no.4, pp.595–603, 1994.
[23] 加藤昇平，世木博久，伊藤英則，“プログラム解析に基づく仮説推論の高速化技法，” 情処学論，vol.35, no.10, pp.2019–2028, 1994.
[24] 高間康史，大澤幸夫，石塚　満，“知識の実行時リフォメーションに基づく仮説推論の高速化手法，” 人工知能誌，vol.10, no.6, pp.913–920, 1995.
[25] A. Satter and R. Goebel, “Consistency-motivated reason maintenance in hypothetical reasoning,” New Generation Computing, vol.15, pp.163–186, 1997.
[26] 越野　亮，林　貴宏，木村春彦，広瀬貞樹，“述語論理知識を扱う全解探索仮説推論の高速化，” 人工知能誌，vol.16, no.2, pp.202–211, 2001.
[27] A.Y. Levy, R.E. Fikes, and S. Sagiv, “Speeding up inferences using relevance reasoning : A formalism and algorithms,” Artif. Intell., vol.97, pp.83–136, 1997.
[28] M. Proietti and A. Pettorossi, “Unfolding-definition-folding, in this order, for avoiding unnecessary variables in logic programs,” Theor. Comput. Sci., vol.142, pp.89–124, 1995.

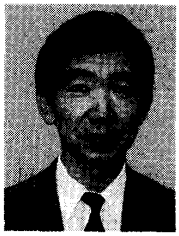
（平成 16 年 2 月 20 日受付，8 月 12 日再受付）

山ノ口　崇

平 12 鹿児島大・工・情報工卒．平 17 金沢大大学院自然科学研究科博士後期課程了．博士（工学）．現在に至る．人工知能システムの検証に興味をもっている．

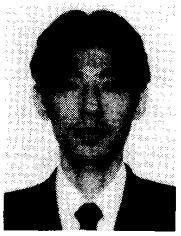
参沢　匡将　（正員）

平 11 金沢大・工・電気・情報卒．平 16 同大大学院自然科学研究科博士後期課程了．博士（工学）．同年金沢工業大学高度材料科学研究開発センター特別研究員．平 17 東京理科大経営学部経営学科助手．現在に至る．マルチエージェント，強化学習に興味をもっている．人工知能学会会員．



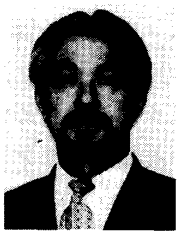
木村　春彦　（正員）

昭 49 東京電機大・工・応用理化卒．昭 54 東北大大学院工学研究科情報工学専攻博士課程了．工博．同年，富士通（株）入社．昭 55 年金沢女子短大講師，昭 56 同短大助教授，昭 59 金沢大・経助教授，平 4 同大・工・電気・情報工学科助教授，平 6 同学科教授，平 16 同大大学院自然科学研究科教授．現在に至る．その間，最適コード変換，プロダクションシステムの高速化の研究に従事．情報処理学会，人工知能学会各会員．



小越　康宏　（正員）

平 13 金沢大大学院自然科学研究科博士後期課程了．博士（工学）．平 9～11 福島女子短大助手．平 11～13 同短大講師．平 13 富山大・工・知能情報工学科助手．平 16 福井大・工・知能システム工学科講師．現在に至る．人工知能の研究に従事．情報処理学会，人工知能学会各会員．



広瀬　貞樹　（正員）

昭 49 富山大・工・電子卒．昭 51 東北大大学院工学研究科修士課程情報工学専攻了．昭 55 同博士課程了．工博．同年，富士通研究所入社．昭 59 神奈川大・工助教授，平元富山大・工助教授，平 10 同知能情報工学科教授．現在に至る．その間，オートマトン・形式言語理論，アルゴリズム解析，計算の複雑さの理論などの研究に従事．情報処理学会会員．