人　工　知　能　76－6
（１９９１．５．２２）

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

鶴田三郎
東京商船大学

石塚　満
東京大学生産技術研究所

発想的仮説生成（abductive hypothesis synthesis）とは，与えられたゴールの証明に必要ではあるが，知識ベース内で不足あるいは欠落している知識を，新たに仮説として生成してくる形態を指す．発想的仮説生成においては知識ベースをコンパイルしておくことが，効率的推論及び知識管理のために不可欠である．そこで，述語論理知識ベースのコンパイル法（ＯＰＣＣ法）を，命題論理を対象として考案した高速コンパイルアルゴリズムを基に開発した．このＯＰＣＣ法を，述語論理知識ベースで発想的仮説推論を行うシステムに適用した結果，instantiationを行う方法と比較した場合，数十倍から数百倍高速で発想的仮説を生成することが可能であった．

A Compiling Method of Predicate Knowledge Base for Efficient Abductive Hypothesis Synthesis

Saburo Tsuruta
Tokyo University of Mercantile Marine,
2-1-6 Etchujima, Koto-ku, Tokyo, 135, Japan

Mitsuru Ishizuka
Institute of Industrial Science, University of Tokyo,
7-22-1 Roppongi, Minato-ku, Tokyo, 106, Japan

A compilation method named "Ordered Predicate Clause Consensus (OPCC) method" directly applies the most general unification and the "Ordered Clause Consensus (OCC) method" to generate prime implicant clauses with variables, instead of applying lifting theorem to the prime implicant clauses generated by ground instantiation. The compiled knowledge-base consisting of these prime implicant clauses is particularly important for efficient abductive synthesis of hypothesis necessary to prove a given goal. In the case of generating prime implicant clauses, the OPCC method is 8-45 times faster than that of simple instantiation method including OCC method.

1．まえがき

完全な知識を対象とした従来の推論に対して，常には正しいとは限らない不完全な知識の範囲迄，対象の知識と推論機構を拡張することは，今後，知識ベースの能力を拡大する上で重要な課題である[1]．論理に基づく仮説推論は，このような不完全な知識を仮説として扱う枠組みであり[2][3]，診断や設計等の適用が可能という実用性も備えた有用な枠組みである．しかしながら，知識ベースが不完全な知識を含むと，一般に非単調推論が必要になり，推論速度が大きな問題となる．

ＣＭＳ（Clause Management System）[4]は，ＡＴＭＳ[5]を論理的基盤上で一般化，拡張したものであるが，発想的仮説生成を知識ベースのコンパイルによって高速に行うという考え方を示している点で重要である[6]．通常の仮説推論では，推論速度の点からあらかじめ可能な仮説を記述しておき，与えられたゴールの証明に必要な無矛盾な部分集合を見い出すような動作を行うが，発想的仮説生成（abductive hypothesis synthesis）とは，与えられたゴールの証明に必要ではあるが，知識ベース内で不足あるいは欠落している知識を，新たに仮説として生成してくる形態を指す．このような発想的推論は，今後の創造的活動の支援に係わる知識システムの基本機能として，非常に重要なものになると考えられる．

ＣＭＳにおいては知識ベースをコンパイルしておくことが，効率的推論及び知識管理のために不可欠である．論理知識ベースをコンパイルするとは，簡単に述べると，ゴールが与えられる前から事前に可能な推論は全て行って導出結果を知識として出しておき，他の知識に包摂されるような冗長な知識を取り除くことを行う．このようにして得られるのが Prime Implicantから成る知識ベースであり，これによってゴールが与えられたときの証明が極めて効率的に行えるようになる．エキスパートシステム的な観点からは，深い知識から浅い知識を生成するといったことに相当する．知識ベースを Prime Implicant節に コンパイルしておけば以後の推論は高速になるが，このコンパイル自身に非常に時間がかかることから，高速コンパイル法が必要になる．

われわれは，命題論理の知識ベースを対象として，順序付け節コンセンサス法（Ordered Clause Consensus method：ＯＣＣ法）と名付けた高速コンパイルアルゴリズムを考案，開発した[7][8]．このコンパイル法では，正リテラルを含む節およびその負リテラルを含む節が存在する双形リテラル（biform literal）の組み合わせ数を算出し，組み合わせ数の少ないリテラルから順序付けて導出を進め，Prime Implicant節を 求める．このＯＣＣ法を用いて，ＣＭＳの枠組みによる発想的推論を行う仮説推論システムを構築し，欠けている知識の発想的生成の高速化に有効であることを確認した[7][8]．

ＯＣＣ法は命題論理を対象としているが，命題論理の表現では変数を扱えないため，知識表現能力は十分でないことが多い．これに対し，述語論理では述語記号の他に，複数の項を引数として記述することが可能であるとともに，変数を用いることが可能であり，命題論理と比較してより高い表現力がある．変数を用いることにより，表現力が高まるとともに，命題論理で表現した知識集合と同じものを圧縮して表現することが可能であり，知識集合の規模の拡大に対処し，知識集合をコンパクトに表現するためにも効果的である．

そこで，命題論理知識ベースについて考察した高速コンパイル手法（ＯＣＣ法）を，関数を含まない１階述語論理を対象に拡張を図ることとした[9]．本論文ではＯＰＣＣ法と名付けた述語論理知識ベースへの拡張方法と，このＯＰＣＣ法を，述語論理知識ベースで発想的仮説推論を行うシステムに適用した結果を記す．

なお，これまでにも知識ベースのコンパイル法を扱った論文はあったが，本論文の論理的枠組みの観点のコンパイルとは異なるものである．たとえば文献(10)はＡＴＭＳのデータ構造とプロダクションシステムのＲＥＴＥネットワークのデータ構造の効率的結合を図るものであるし，文献(11)

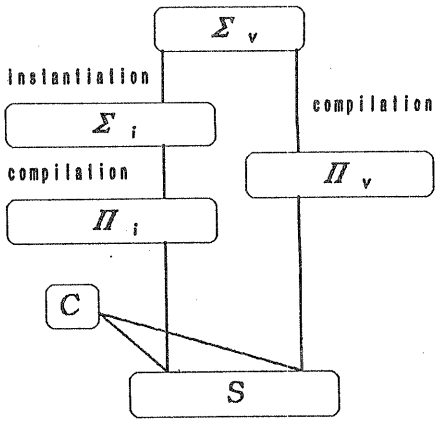
は論理的枠組みとは離れて，エキスパートシステム的な観点で深い知識から浅い知識を生成しようとするものである．

2．述語論理知識ベースのコンパイル法

2．1　順序付け節コンセンサス法の拡張[9]

述語論理を対象とするための方法として，述語論理の論理式の中の変数に，エルブラン空間の要素を基礎代入する方法（instantiation）を挙げている論文が多い[4][12]．この場合，命題論理に対するコンパイル法を適用できることになるが，関数を含まない論理式に限定しても，空間が節の数，定数の数，リテラル中の項の数に比例した大きさになるという問題があり，現実的でない．

論理式においては，代入が適用可能であれば，束縛変数を書き換えることが可能である[13]．このことは自由変数のみならず束縛変数についても，変数に変数を代入することが可能であることを意味している．そこで，変数を当初から全部具体化（instantiate）することを考えず，導出に際して，変数を含まない基礎代入ではなく，変数を代入できるものについては変数を代入し，変数を含んだ述語論理節について，Prime Implicant節を 導出することを考える（Fig. 1）．この考え方は，基礎例（具体例）である Prime Implicant節に対し，



Σ_v　axioms with variables (predicate knowledge-base)
Σ_i　instances of axioms
Π_v　prime implicants with variables (compiled knowledge-base)
Π_i　prime implicants of instances (compiled knowledge-base)
C　query
S　minimal supports satisfying the following relation ;

$\Sigma \vdash S \vee C$, $\Sigma \nvdash S$, $\Sigma \nvdash S'$, $\Sigma \nvdash S' \vee C$　for any subset S' of S .
$\sim S$ becomes an abductively generated hypothesis for a given query C .

Fig. 1　Two compilation methods for predicate logic knowledge-base ; instantiation method (left line), in which the Ordered Clause Consensus(OCC) method for propositional logic is applied, and Ordered Predicate Clause Consensus(OPCC) method (right line).

持ち上げ定理（lifting theorem）[14] を適用したことに相当し，変数付きで表された Prime Implicant節を導出することになる．持ち上げ定理を適用すると，エルブラン空間の定数の代入による，節集合の規模の増大を避けることが可能となる[15]．すなわち，述語論理による公理節集合に対して，あらかじめ全部のinstantiationを行わないことにより，Prime Implicant節 集合の規模の増大を避けることが可能となる．

この考え方により，ＯＣＣ法の適用範囲を述語論理に拡張する．本論文においては，述語論理節を構成するリテラルの中の項としては，定数と変数のみを考え，関数を考えないこととする．変数の代入を優先する代入をθとし，任意の単一化代入をλとすると，関数を対象としないことから$\lambda=\theta\cdot\sigma$となる代入$\sigma$が存在し，この変数を許容する方法は，この範囲で最汎単一化代入法（most general unifier method）である． この最汎単一化代入法を用いたＯＣＣ法の拡張版を，ここではＯＰＣＣ法（Ordered Predicate Clause Consensus method） と呼ぶことにする．ＯＰＣＣ法においては，再帰的な節は対象としていない．リテラルの中の項としての変数に，導出に先だって定数を代入し，この基礎例を用いてコンパイルを行う方法を，以後 instantiation法と呼ぶことにする．

なお，Fig. 1にも示されているように，Prime Imlicant節 から成るコンパイルされた知識ベース（Π_vやΠ_1）が 得られていると，証明のゴールとなる質問節Ｃが与えられると，Ｃを含む Prime Implicant節（Π）を検索し，Ｃとの差分をとることによって（Ｓ＝Π－Ｃ），Minimal Support節（Ｓ）を求めることができる[4]．このＳの否定～Ｓが質問節（Ｃ）を証明するために必要な不足していた知識であり，発想的に仮説として生成されることになる．Ｓあるいは～Ｓが空集合の時には仮説生成は不要であり，質問節（Ｃ）が元の知識ベースから証明できる論理的帰結であることが判明する．

２．２　述語論理への拡張に当たっての検討事項

述語論理で表現された節集合について導出を行うためには，次の３項目について検討を行う必要がある．

①節中の変数に代入を行い，導出のための単一化を行うこと．

②単一化を行った節同士の間で導出を行うこと．

③節について因子化を行うこと．

さらに，ＯＣＣ法を適用するにあたっては，上記３項目に加えて次の２項目について検討する必要がある．

④包含関係による節の削除の方法．

⑤順序付けの方法．

次章では，上記５項目について検討を行うこととする．

３．述語を対象としたコンパイル法（ＯＰＣＣ法）

３．１　ＯＰＣＣ法の概要

（１）単一化代入

述語論理の導出過程においては，代入が適用可能であれば，変数を変更することが可能である．単一化代入を行うにあたって，まず導出の親節となる二つの節が共通の変数を持つ時には，一方の節の変数名を変え，共通の変数を持たないように，改名代入を行う必要がある．次いで，変数に項を代入するが，変数を代入することが可能な変数については，変数による単一化を行う．このように項として変数を許容することにより，全部をinstantiateする方法と比較して，問題空間の拡大を制限することが可能となる．

（２）導出

ＯＣＣ法においては，二項導出を用いている．ＯＰＣＣ法における述語論理を対象とした二項導出の方法は次の通りであり，通常の最汎単一化による導出と同じである．

①項として定数を含む節のみに関して導出を行う場合，命題論理と同じである．

②項として定数を含む節と，変数を含む節とに

関して導出を行う場合，変数に定数を代入して導出を行う．

③項として変数のみを含み，定数を含まない節同士に関して導出を行う場合，変数には変数を代入して導出を行う．

例　{p（X）Vq（Y），～p（Y）}の場合

共通の変数があるため，まず第2節に対して改名代入　θ1＝{Y：＝Y2}を適用すると，

{p（X）Vq（Y），～p（Y2）}となる．

次いで，代入θ2＝{Y2：＝X}を適用すると，q（Y）が導出される．

（3）因子化

導出節の中のリテラルとして，同じ述語記号を持つものが存在することがあり，変数の代入を許容する場合には，因子化を検討する必要がある．ある節の中に，単一化可能なリテラルが存在する場合には，この一つの節の中で変数に代入を行い（因子化），因子（factor）を生成することができる．リテラルの数や変数の項数が多い場合には，因子が多数生成される場合がある．

例　{p（a）Vp（X）Vp（Y）}の場合，

θ1＝{Y：＝a}を適用すると，因子は{p（a）Vp（X）}となる．

θ2＝{X：＝a}，　θ1＝{Y：＝a}を適用すると，因子は{p（a）}となる．いづれも元の節の因子である．

（4）包含関係による節の削除

命題論理の場合は節の包含関係により，他に包摂される節を簡単に削除する事ができたが，述語論理の場合には，変数の項と定数の項との関係にも考慮し，節集合の中から他の節に包含される節（冗長な節）を削除する．ＯＰＣＣ法での，冗長な節を削除する方法を次に示す．

①項として定数のみを含む節間の場合，命題論理と同じである．

②項として定数を含む節と，変数を含む節との場合，述語記号が同じであれば，変数の項を持つ方をより一般的であるとし，冗長な節を削除する．

③項として変数のみを含み，定数を含まない節同士の場合，命題論理と基本的に同じであり，冗長な節を削除する．

この考え方は包摂（subsumption）の考え方に近いが，定義が「節Ａは，節Ｂの部分節である節Ａの基礎例が存在するとき，節Ｂを包摂する[15]」であることから，上に示した考え方の②と包摂は同じであることになる．ここでは，命題論理で用いた①の方法や，変数のみの場合に拡張した③の方法も併せて用い，冗長な節の削除を行う．

（5）順序付けの方法

述語論理を対象としているＯＰＣＣ法で，正式に順序付けを行うにあたっては，定数と変数とを区別して組み合わせ数を計算する必要がある．しかしながら，算出のオーバーヘッドを軽くするために，定数を含む節と，変数を含む節とを区別せず，項数が等しく，同じ述語記号を持つリテラルのみに注目して分類し，組み合わせ数を概算して順序付けを行う簡略法の方が現実的であり，ここではこの簡略法を用いた．

3．2　述語を対象としたコンパイル法の効果についての検討

（1）知識の1つの節の中のリテラルの項がすべて同じ変数である場合

instantiation法では，質問節が与えられた時点で，質問節に含まれる定数もエルブラン空間に含めて知識ベースの instantiationを行い，この結果である基礎例から構成される節集合に対して，ＯＣＣ法を1回適用しPrime Implicant節を求める．このPrime Implicant節と質問節との集合演算により，Minimal Support節を 生成することになる．

ＯＰＣＣ法では，コンパイルを1回行うことにより，変数の項を持ったリテラルによる節の形でPrime Implicant節を得る．このコンパイルした節集合に対して，1つの定数のみを含む質問節を与えた場合，各Prime Implicant節に対する1回の

instantiateと集合演算とによりMinimal Support節を生成することができる．この範囲では，コンパイルの計算量，Prime Implicant 節の数，Minimal Support節の 数，生成に要する計算量のいずれも，instantiation法と ＯＰＣＣ法とで同程度である．

しかしながら，引き続いて異なる定数を持った質問節について，Minimal Support節を生成する場合，instantiation法の場合は 先に行ったコンパイルは，このMinimal Support節を生成するためには 何の効果も持たず，ほとんど同様なコンパイルをもう一度繰り返し行う必要がある．これに対し，ＯＰＣＣ法ではPrime Implicant節を 新たに導出する必要が無いため，もう一度コンパイルする必要は無く，質問節中の定数に対応する１回のinstantiationと，集合演算とのみによりMinimal Support節を 生成することができるため，計算量はＯＰＣＣ法の方が大幅に少なくなる．

（２）知識の１つの節の中のリテラルの項がすべて変数で２種類以上存在する場合

instantiation法では，定数の組み合わせ数に比例して公理節集合が拡大することにより，導出のための計算量が大きくなるとともに，Prime Implicant節も 多くなり，Minimal Support節を生成するための集合演算量も多くなる．最大では定数の組み合わせ数をｎ，定数の数をｋ，変数の数をｍとすると，$n = k^m$ となるが，同じ節も存在するため，可能な組み合わせ数は通常これより少ない．

これに対しＯＰＣＣ法では，Prime Implicant節の 数は instantiation法に比較した場合，大幅に少なく，また Minimal Support節を生成する場合も，Prime Implicant節の数自体が少ないことから，計算量は大幅に少なくなる．

（３）知識のリテラル中の項として定数と変数とが存在する場合

節中の定数が１種類だけの場合，質問節中の定数が公理節中の定数と同じの場合には，instantiation法の 計算量と，ＯＰＣＣ法のとはほぼ同程度である．

定数が異なる場合には，instantiation法では基礎代入の数が増加することにより，Prime Implicant節の 数も増加し，その分計算量が増加する．

変数の数が増加すると，instantiation 法では，節中の変数が１種類だけの場合でも，定数の数に比例して公理節の数が増加するし，さらに変数の数にも比例して公理節の数が増加する．

これに対しＯＰＣＣ法では，因子により元の節を包含関係から削除できるとは限らないにしても，公理節の数は instantiation法と比較して少なく，この傾向は定数の数や変数の数の増加に伴い顕著となる．特に変数が多い場合に，ＯＰＣＣ法は有効である．

（４）知識のリテラル中の項がすべて定数の場合

項として変数を含んだリテラルが存在しないため，instantiationを行う必要はない．

３．３　述語論理を対象とした発想的仮説生成を行う推論システムの評価

（１）評価方法

述語論理を対象としたシステムを，Ｐｒｏｌｏｇで構築した．システムをＰｒｏｌｏｇを用いて開発したのは，単一化代入，改名代入等で必要となる変数を変数に代入する処理について，Ｐｒｏｌｏｇがこれを可能とする機能を持っているためである．この述語論理を対象としたシステムを評価するために，まず述語を対象としたことによる負荷の変化を計測し，次いで Prime Implicant節および Minimal Support節の生成のためのＣＰＵ時間に関する，ＯＰＣＣ法と instantiation法との比較を行った．

（２）述語を対象としたことによる負荷の変化

述語論理を対象としたことにより，リテラルが１つ以上の項を含むことになったことによるシステムの負荷の増大の傾向を見るため，乱数を用いて発生させた項を持たない（項の数が０に相当する）命題論理節集合と，リテラル中の項の数が１と２である述語論理節集合とについて，また実際の知識を表現した節集合について，Prime

Implicant節の 導出時間の比較を行った．なお，リテラル中の項の数が２のものについては，変数を共有する場合と，共有しない場合について行った．

命題論理を対象としたシステムを，述語論理を対象とするように拡張したことにより，比較した例においては，乱数による節集合の場合で最大約1.5倍，実際の知識による節集合の場合で最大約1.1倍へと，ＣＰＵ時間が多少増加する傾向があった．しかしながら，その増加の程度はそれほど著しいものではなかった．

（３）代入方法の比較

instantiation法 とＯＰＣＣ法とを，ＣＰＵ時間，および公理節の数について比較を行った．instantiation 法においては，次に示す方法により基礎代入を行った後，ＯＣＣ法を用いて Prime Implicant節の導出を行った．

①公理節集合中および質問節中の定数を全部探す．

②変数を含む論理式中の変数に定数を代入し，可能なすべての基礎代入を行う．

③ＯＣＣ法により，Prime Implicant節の導出を行う．

④質問節に対しMinimal Support節の生成を行う．

実際の知識を表現した公理節集合を対象とした Prime Implicant節の生成時間をTable１に，また，Minimal Support節の生成時間をTable２に示す．

乱数により発生させた知識集合を対象とした場合，節中の変数の数が１で，質問節中の定数も１種類の場合には，二つの方法の間にはほとんど差は見られなかった．しかしながら，節中の変数の数が２で，質問節中の定数も２種類を許容する場合には，instantiation法においては，基礎代入された公理節の数が４倍となり，Prime Implicant節を生成するためには，乱数を用いていることにより，この４倍となった公理節の中のリテラルについて，ほとんどすべての組み合わせを計算する必要があるため，当初の公理節の数が16の場合においても12000ｓｅｃ以内ではPrime Implicant節を求めることができなかった．

実際の知識を表現した節集合を対象とした場合，乱数の場合ほど極端にはならなかったが，ＯＰＣＣ法の方が単純なinstantiation法に対して，Prime Implicant節の生成に関しては，８－45倍高速であった．同じ内容を命題論理で表現し，幅優先法を適用した場合と比較すると，６－200倍もしくはそれ以上高速であり，公理節の数が多い場合に特に効果的である．実際の知識を表現した公理節に対して，単純に基礎例を生成すると，例えば船舶航行エキスパートシステム[16]において自船が自船自身を避けるといった，変数の間に有り得ない組み合わせが生成されるため，組み合わせに制限を設けた例についても，Prime Implicant節の生成時間の計測を行った．この場合，公理節の数が少なくなったことに比例してＣＰＵ時間も小さくなり，高速化の程度は Prime Implicant節の生成に関しては，３－７倍となった（Table１）．

（４）Prime Implicant節の数と導出時間

instantiation法では，初めに全ての基礎代入を行うことにより，導出に利用されない節も含めて多数の節が生成されるため，Prime Implicant節導出のＣＰＵ時間が大きくなり，Prime Implicant節の数も多くなった．これに対し述語を対象としたＯＰＣＣ法では，変数を含んだ項を許容しているため，Prime Implicant 節の数は多くはならず，そのため導出に要するＣＰＵ時間も大きくはならなかった．

実際の知識の節集合を述語論理を用いて表現すると，命題論理と比較した場合，公理節数の総数が減少し，これに比例して同じ意味を持つ Prime Implicant節を導出するＣＰＵ時間が減少した．今回用いた例においては，リテラル中の項の数は２であるが，たとえば命題論理において節数が50で表現される知識を，述語論理ではこれより少ない６割の31の節で表現することが可能である．この時には Prime Implicant節を導出する時間は，命題論理にに対してＯＣＣ法を適用し高速化を図っても，約 2.5倍の時間が必要であった（Table１）．

命題論理を対象としたシステムでは，当初の節数が２倍になると時間は約６倍に，節数が４倍に

なると時間は約20倍になり，また当初のリテラル数が２倍になると時間は約10倍に，リテラル数が４倍になると時間は約80倍になる[7][8]．このことから，述語論理を用い，変数を持った Prime Implicant節を許容して節の数およびリテラル数を減少させることは，Prime Implicant節 を導出するＣＰＵ時間を減少させるために，非常に効果があるということができる．

（５）Minimal Support節の生成時間

Prime Implicant節に ついて検討したものと同じ節集合について，Minimal Support節 生成時間の比較を行った．実際の知識を表現した節集合を対象とした場合，単純な instantiation法に対して，ＯＰＣＣ法の方が２－15倍高速であった．組み合わせに制限を設けた例については，公理節の数が少なくなったことに比例してＣＰＵ時間も小さくなり，高速化の程度は約１－６倍となった（Table２）．いずれも知識ベースの規模が大になるにつれて，ＯＰＣＣ法の高速化の倍率が増す．

instantiation法では，多数のPrime Implicant節が生成されるため，Minimal Support節生成のためのＣＰＵ時間が大きくなるとともに，Minimal Support節の数もinstantiationを行っているため多くなった．これに対し，あらかじめ基礎例を生成しないＯＰＣＣ法では，変数を含んだ項を許容しているため，Prime Implicant節の数は多くはならず，そのため Minimal Support節生成に要するＣＰＵ時間も大きくはならなかった．

しかしながら，instantiation法の結果に顕著に見られるように，公理節の数が大きくなった時に，Minimal Support節 の生成時間が急激に増大する傾向が見られる．これはシステムを汎用のものとしたために，常に節の包含関係の検討を行うことによるものであり，Prime Implicant節 集合が大

Table１　Computational time required to generate Prime Implicant clauses in the case of practical knowledge-base

Propositional Clause Knowledge-base			Predicate Clause Knowledge-base					
	ＯＣＣ	Breadth-first Method	ＯＰＣＣ		Simple Instantiation ＋ＯＣＣ		Restricted Instantiation ＋ＯＣＣ	
Number of Axiom Clauses	CPU Time (sec)	CPU Time (sec)	Number of Axiom Clauses	CPU Time (sec)	Number of Instanti-ated Axiom Clauses	CPU Time (sec)	Number of Instanti-ated Axiom Clauses	CPU Time (sec)
１２	8.1	55.5	１１	9.4	４４	78.9	２２	25.9
３２	143.9	14555.	２２	70.6	８８	3625.2	４４	540.2
５０	257.5	above 15000	３１	100.9	１２４	4447.7	６２	755.0

きい場合に顕著である．質問節がユニット節に限定される場合，包含関係の検討は不必要であり，さらに高速化を図ることが可能である．

4．むすび

述語論理知識ベースの高速コンパイルを可能とする，ＯＰＣＣ法（Ordered Predicate Clause Consensus method）について記した．これは命題論理知識ベースの高速コンパイル法である順序付け節コンセンサス法（Ordered Clause Consensus method，ＯＣＣ法）の，述語論理への拡張となっている．本論文では発想的仮説生成を行う推論システムの効率化を主たる目標にして記したが，一般の述語論理知識ベースでの推論の効率化としても有効な手法である．述語を対象としたＯＰＣＣ法と，単純に全部の基礎例を生成し，この基礎例についてＯＣＣ法によるコンパイルを行う方法（instantiation法）とを比較した結果，ＯＰＣＣ法は問題空間の拡大を抑制する述語論理を対象とした知識ベースの高速コンパイルを可能にしている．また発想的仮説生成を行う推論システムにおいて，高速で Minimal Support節を生成することを可能としている．

論理知識ベースのコンパイルは，推論の高速化のために極めて有効な手段であるが，知識ベースの規模が大きくなると完全に Prime Implicant節を求めることは，計算時間の点，メモリ容量の点で現実的でなくなる．そこで，可能な部分まで事前にコンパイルをしておく論理知識ベースの部分コンパイル法，及び証明のゴールとなる質問節が与えられてから部分コンパイル知識ベースから高速に解を求める推論機構が必要になると考え，研究を進めている[17]．知識ベースのコンパイル法

Table 2　Computational time of Minimal Support clauses in the case of practical knowledge-base

Propositional Clause Knowledge-base		Predicate Clause Knowledge-base					
ＯＣＣ		ＯＰＣＣ		Simple Instantiation ＋ＯＣＣ		Restricted Instantiation ＋ＯＣＣ	
Number of Axiom Clauses	CPU Time (sec)	Number of Axiom Clauses	CPU Time (sec)	Number of Instanti-ated Axiom Clauses	CPU Time (sec)	Number of Instanti-ated Axiom Clauses	CPU Time (sec)
12	0.7	11	0.7	44	1.3	22	0.9
32	9.0	22	2.7	88	44.5	44	16.3
50	10.9	31	3.1	124	49.5	62	18.8

は学習機能と推論の高速化という点で深い関係がある．コンパイル法がコンピュータ技術の側面からの高速化へのアプローチであるのに対し，学習は人工知能的側面からのアプローチであると言える[18]．

今後，知識ベースは高次推論を取り入れて能力の拡大を図る必要があるが，高次推論になればなる程一般に必要とする計算量は増大する．従って，高速の推論メカニズムを併せて開発しないと実用にならず，本論文に記したようなコンパイル技術は学習機能と同等に重要になると言える．

参考文献

（1）石塚　満：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）Pool，D.，Aleliunas，R.，Goebel，R.：Theorist：A Logical Reasoning System for Defaults and Diagnosis，in The Knowledge Frontier：Essays in the Representation of Knowledge（N.J.Cercone，G.MaCalla Eds.）Springer-Verlay（1987）.

（3）國藤　進：仮説推論，人工知能学会誌，Vol.2，No.1，pp.22-29（1986）.

（4）Reiter，R.，de Kleer，J.：Foundations of Assumption-based Truth Maintenance Systems：Preliminary Report，Proc.AAAI-87，pp.183-188（1987）.

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

（6）鶴田三郎，石塚　満：不完全な知識が係わる知識システムへのＣＭＳの応用に関する研究，第３回人工知能学会全国大会，pp.179-182（1989）.

（7）鶴田三郎，石塚　満：命題論理知識ベースのコンパイル法，情処学会人工知能研資料，90-AI-70（1990）.

（8）鶴田三郎，石塚　満：発想的知識生成のための命題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（1991）.

（9）鶴田三郎，石塚　満：述語論理知識ベースのコンパイル法，第４回人工知能学会全国大会，pp.81-84（1990）.

（10）井上克己，太田好彦：仮説推論システムAPRICOT/0における知識コンパイル，人工知能学会研究資料，SIG-KBS-8805，pp.51-60（1989）.

（11）山口高平，溝口理一郎，田岡直樹，小高浩，野村康雄，角所　収：深い知識に基づく知識コンパイラの基本設計，人工知能学会誌，Vol.2，No.3，pp.333-340（1987）.

（12）Poole，D.L.：On the Comparison of Theories：Prefering the Most Specific Explanation，Proc. 9th IJCAI，pp.144-147（1985）.

（13）松尾文碩：述語論理による知識の表現と推論，知識の表現と利用（上野春樹，石塚　満（共編）），第５章，オーム社（1987）.

（14）有川節夫，原口　誠：述語論理と論理プログラミング，オーム社（1988）.

（15）Wos，L.，Overbeek，R.，Lusk，E. and Boyle J.：Automated Reasoning Introduction and Applications，Prentice-Hall Inc.（1984）.

（16）鶴田三郎，松村尚志，稲石正明，今津隼馬，杉崎昭生：船舶航行エキスパートシステムの基礎研究－衝突回避エキスパートシステムについて－，日本航海学会論文集，No.77，pp.133-139（1987）.

（17）遠藤裕明，鶴田三郎，石塚　満：知識コンパイルの部分的適用による高速仮説推論システム，情報処理学会第４２回全国大会7F-1（1991）.

（18）石塚　満：人工知能の夢への接近，情報処理，Vol.32，No.1，pp.7-9（1991）.