人 工 知 能 70－6
（1990．5．9）

命題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

論理的推論の高速化に関する技術としては，ＴＭＳ，ＡＴＭＳ，ＣＭＳ，ＣＳＰ等が発表されている．この中で，Reiter他の提案したＣＭＳ（Clause Management System）は，論理基盤上でＡＴＭＳを一般化，拡張したものとして位置付けられ，知識ベースのコンパイルにより，推論の高速化を図るとともに，無矛盾性の管理および欠落した知識の発想を可能とする枠組でもある．本研究は，命題論理を対象とした仮説推論に，この枠組を応用するための，知識ベースのコンパイル法を開発したものである．単純な幅優先法と比較した場合，今回開発した方法により約５～９０倍以上高速でPrime Implicant節を導出することが可能であった．

A Compiling Method of Propositional Knowledge Base

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

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

Truth maintenance systems including TMS, ATMS, CMS(Clause Management System) have been developed for the efficient management of hypothesis which is a type of the incomplete knowledge. Among these systems, CMS has a logical structure and suggests the important properties of knowledge compilation and abductive function. This paper proposes a fast and efficient algorithm for the generation of the prime implicant clauses which allows the compilation of propositional knowledge-base. The knowledge compiling speed of this system is 5 to 90 times faster than that of simple implementation using the basic function of Prolog.

1．はじめに

現在，連想，類推，帰納推論，発想などの高次人工知能機能を実現し，利用することがＡＩ研究の大きな課題となっている．われわれは「完全な知識に加えて不完全な知識も含めて知識ベースを構成し，これを操作することにより，高次人工知能機能を実現する」という視点から，次世代知識ベースの構成を検討し，研究を進めている(1)．不完全な知識を含む知識ベースを操作する場合，常には正しいとは限らない知識については，無矛盾性を確認する必要があり，また欠落部分を含む知識については，欠落した知識を発想する必要がある．無矛盾性を確認する計算量や，発想のために新しい組み合わせを生成する計算量は，問題空間の拡大に伴い指数的に増大することから，これらの機能を実現するためには，高速化技術の開発，研究が必要となる．論理的推論の高速化に関する技術としては，ＴＭＳ(2)，ＡＴＭＳ(3)，ＣＭＳ(4)，ＣＳＰ(5)(6)等が発表されている．ＡＴＭＳやＣＳＰは矛盾する空間の探索を避けることにより，推論の高速化を図っており，ＡＴＭＳはＣＳＰよりも，より一般的な問題に対応することが可能である(7)．Reiter他の提案したＣＭＳ（Clause Management System）は，論理基盤上でＡＴＭＳを一般化，拡張したものとして位置付けられ，知識ベースのコンパイルにより，推論の高速化を図るとともに，無矛盾性の管理および欠落した知識の発想を可能とする枠組でもある．

無矛盾性の管理および欠落した知識の発想を可能とする，Reiter他の示したＣＭＳという枠組は，仮説推論に有効であると思われるため，われわれは基本的なＣＭＳを実現し，有効性と実際の知識システムに応用するにあたっての問題点を検討した(8)．ＣＭＳは，知識ベースのコンパイルにより，導出を行なうことなく，Minimal Support節を Prime Implicant節との包含関係から直接高速生成することを可能とする枠組ではあるが，これを利用して実用的な仮説推論システムを実現するためには，Prime Implicant節をあらかじめ導出するという，知識ベースのコンパイルが重要となる．しかしながら，Reiter他は Prime Implicant節を用いた推論の高速化の可能性を示しているものの，そのために必要となるコンパイル技術に関しては検討を十分に行なっていなかった．本論文は，命題論理を対象とした仮説推論に，Reiter他の示したこの枠組を応用するための，知識ベースのコンパイル法の開発，およびコンパイルした知識を利用した Minimal Support節の高速生成法の実現を目的とするものである．今回開発した Prime Implicant節の高速導出方法の基本的な部分は，論理関数において積項を用いて主項を生成する方法の一種である，Tison法を変形したものに相当するが，節を対象としている点が大きく異なる点である．

2．基本的な枠組

2．1　ＣＭＳ（Clause Management System）の概要

ＣＭＳは，一般の命題論理節を対象とし，公理節集合（Σ）と，問題解決器から送り込まれた質問節（Ｃ）とから，Minimal Support節（Ｓ）を生成することにより，質問節の論理的帰結性や，無矛盾での充足可能性を検討する枠組である．論理的帰結性を問う質問節の否定に対して，空節ではない Minimal Support節が生成された場合（$\Sigma \vdash \sim C \rightarrow S$，$S \neq \{\}$）が，質問節が公理節集合から論理的に帰結されない場合であり，Minimal Support節の否定は証明するために欠けている知識に相当する（$\Sigma \vdash \sim S \rightarrow C$）．すなわち欠けている知識は連言から成り立つ論理式で表現される．また，Minimal Support節が空節である場合（$S = \{\}$）は，ＣＭＳ中の公理節集合によって質問節を論理的に帰結することが可能で，欠けている知識が無い場合に相当する．充足可能性を検討する場合，連言論理式で表現された質問の否定としての質問節に対して，Minimal Support節として空節が生成されない場合が，無矛盾で充足可能

である場合に相当する．これらのことから，この枠組は，常には正しいとは限らない知識や，欠落部分を含む知識等の，不完全な知識を含む知識ベースを操作する必要のある，仮説推論システムに適していると思われる．

ここで Minimal Support節は，次のように定義される．

① $\Sigma \vdash S \vee C$

② $\Sigma \nvdash S$

③ Sの部分集合S'について，
$\Sigma \nvdash S'$，$\Sigma \nvdash S' \vee C$

2．2 Prime Implicant節導出の高速化の必要性

Minimal Support節を生成する方法としては，節をそのまま登録して，質問に応じて Minimal Support節を計算するインタプリタ形式によるアプローチと，あらかじめ Prime Implicant節を導出しておくコンパイル形式によるアプローチとがある．Minimal Support節は必ずしも空節でないため，空節が生成されない場合には，基本的には全部の Minimal Support節を導出する必要がある．文献(8)においては，インタプリタ形式を採用したが，この形式においては，公理節集合中の節数およびその節中のリテラル数が多い場合に，導出節が急激に多くなり，Minimal Support節を生成するための計算量が急激に大きくなる問題がある．

コンパイル形式は，公理節集合からあらかじめ導出した Prime Implicant節と，質問節の包含関係から，直接 Minimal Support節を高速に生成する方法である．本論文においては，Minimal Support節の生成の高速化を図るための方法として，効果の大きいと思われる，コンパイル形式を対象として検討することにした．コンパイル形式で高速化を図るためには，Prime Implicant節の導出の高速化を図ることが必要となる．

ここで，Prime Implicant節（Π）は次のように定義される．

① $\Sigma \vdash \Pi$

② Πの部分集合Π'について，$\Sigma \nvdash \Pi'$

3．導出の高速化

3．1 高速化アルゴリズム

導出の高速化のために通常用いられる，意味導出，支持集合導出，線形導出等の戦略は，空節の導出を目的としているために，今回のように Minimal Support節が必ずしも空節でない場合には，高速化の戦略としてそのまま利用することはできない．幅優先戦略等による全解探索は，効率が悪く，特に公理節が多くなった場合には実用的ではない．

そこで，公理節集合中のリテラルのうち，節集合中に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）に注目し，この双形リテラルを利用した Prime Implicant節導出の高速化の方法（順序づけ節コンセンサス法）を開発した．アルゴリズムを下に示す．

順序づけ節コンセンサス法

①すべての双形リテラルについて，正リテラルを含む節およびその補リテラル（complementary literal）を含む節の間の組み合わせ数を算出する．

②導出に用いられていない双形リテラルのうち，①で計算した組み合わせ数の最も少ないリテラルを選ぶ．

③選択されたリテラルについて導出を行なう．

④他の節を包含する節を消去する．

⑤すべての双形リテラルについて導出が終了していない場合，②に戻る．

この方法では，公理節集合に含まれる双形リテラルについて，それぞれただ1回導出を行なえばよいため，導出の終了が明らかである．また他の

節を包含する節については毎回消去を行ない，対象としている節の減少を図るとともに，導出に先だって組み合わせ数を計算し，導出を行なう双形リテラルの順序づけを行なうことにより導出回数の減少を図り，導出の高速化を図っている．

3．2　論理関数における主項の生成方法との関係

今回の知識ベースは節を対象としており，節集合は和積形で表現される．和積形は積和形に変換することが可能であり，論理関数においては，主項は変数の積の形で表現されるため，論理関数の主項の生成方法の双対な方法が，節を対象としたPrime Implicant節の導出方法となりうる．主項の生成方法としては，図表による方法[9][10][11]，和積形の展開に基づく方法[12][13]，共有展開に基づく方法[14][15]等がある．双対な方法による Prime Implicant節の導出方法では，公理が節集合で示された場合，これを積和形に変換した関数が元の関数の偽に対応するとし，この偽の関数の主項を求め，これを元の節集合の Prime Implicant節に変換することになる．ＣＭＳの提案者であるReiter他は，和積形の展開に基づく方法の１種であるSlagle-Chang-Leeの方法[12]をその候補として示唆しているが，和積形に展開するためには，偽である積和形から真である積和形を求める必要が有り，公理節集合の偽集合はもとの集合よりも大きい場合が少なくないこと，また節集合に無い節が真か偽か不明であることにより，この方法の応用は難しい．

共有展開に基づく方法（コンセンサス法）の１種である Tisonの方法[14]は，積項を対象としているため，偽である積和形から真である積和形を求める必要が無く，Prime Implicant節を導出するための双対な方法として利用することが可能である．Tisonの方法により双対問題として Prime Implicant節を生成する場合，双対問題への変換と，これから求めた主項を元の問題の Prime Implicant節に逆変換する必要がある．節を対象とした今回の導出方法は，積項と節との違いは有るものの，Tisonの方法によく似ており，コンセンサス法の１種であるということができる．順序づけ節コンセンサス法は，出発点の違いにより，双対問題に変換すること無く，節に対して直接導出を行なっている点，および順序づけを行なっている点が，Tisonの方法と異なっている点であり，より速く Prime Implicant節を導出することを可能としている．

論理関数においては，変数の数が多くなった場合，これに比例する形で，１つの項の中の変数の数が多くなるのが，普通のケースであると思われるが，実際の知識ベースにおいては，公理節集合中のリテラルの数が多くなった場合でも，リテラルの数の多い節は非常に稀な例であり，たとえば船舶の航行に関するプロダクションルールの本体部の条件の個数は約３であり[16]，１つの節の中のリテラルの数はあまり多くならないと思われる．この意味で，実際の知識ベースを対象とする場合，節集合中のリテラルの数が多くなった場合でも，計算速度の低下は論理関数におけるほど顕著ではないと思われ，コンセンサス法の効果がより期待できる．

3．3　Minimal Support節生成の高速化

Reiter他は質問節（Ｃ）が Prime Implicant節（Π）に包含される場合，Minimal Support節（Ｓ）を簡単に生成できることを指摘している．Π－ＣによりＳを求めるこの方法は，質問節がユニット節である場合，MinimalSupport節の生成を高速化する有効な方法である．（ここでΣがホーン節のみで構成され，質問節Ｃがユニット節である場合が，ＡＴＭＳに相当する．）

質問節がユニット節でない場合には，Minimal Support節ではない節が生成されることがある．この場合，求められた Minimal Support節の候補の間で包含関係を検討し，Minimal Support節の生成を行なう必要がある．この方法は質問節を構成するリテラルについて，その補リテラルを用い

て導出を行なうことと同意であり，次のように拡張することが可能である．C≠Πである場合には，C'⊆ΠであるCの部分集合C'を求め，このC'に対して前述の方法を適用することにより，高速化を図ることが可能である．このようにPrime Implicant節があらかじめ導出されている場合，質問節の否定を用いて導出を行なうことなく，集合演算によりMinimal Support節の生成の高速化を図ることが可能である．

この方法は無矛盾での充足可能性を検討する場合にも応用することが可能である．すなわち，質問の連言論理式の否定である質問節を用いて集合演算を行なうことにより，導出を行なうことなくPrime Implicant節を用いて，Minimal Support節を生成することが可能であり，無矛盾性の検討を高速で行なうことが可能となる．質問の連言論理式を構成する各リテラルについて導出をしたことと同じ意味であり，質問を追加充足する場合には，生成されたMinimal Support節が新しいPrime Implicant節となる．

4．コンパイル法による仮説推論システムの構築と動作

4．1　システムの構成

構築した仮説推論システムは，命題論理を対象としており，管理部，データ変換部，データベースおよび導出部から構成される．（Fig. 1）

①管理部（Management Module）

問題解決器（Reasoner）から質問節を受け取り，欠落した知識の仮説として，もしくは無矛盾での充足可能性の判定結果として，Minimal Support節を問題解決器に戻す．また，データ変換部に対して問題解決器のデータベースからのデータの取り込みの指示，導出部に対してPrime Implicant節の導出，もしくはMinimal Support節の生成指示を行なう．

②データ変換部（Data Converter）

問題解決器のデータベースから公理節集合を取り込み，データの表現形式をシステム用に，公理集合名，節等の項よりなる複合項の形に変換する．

③データベース（Data Base）

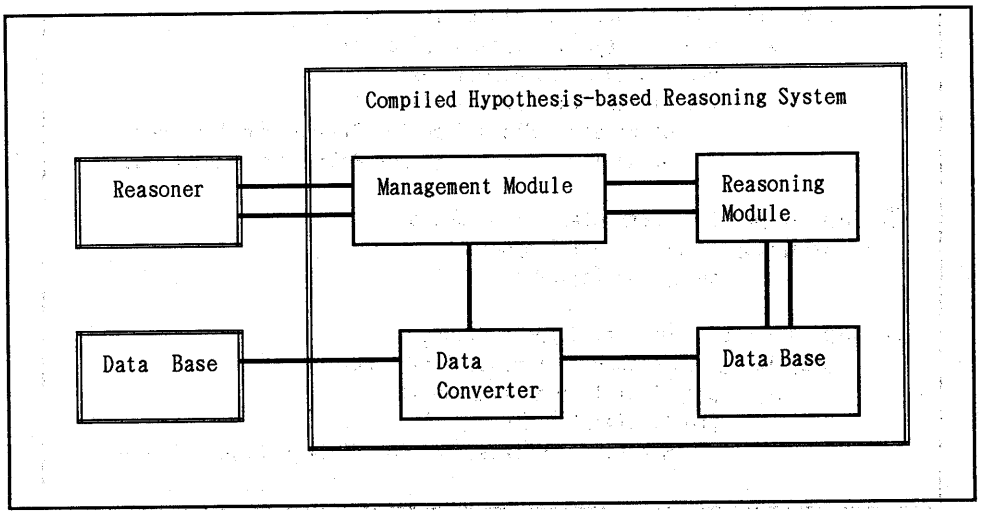

Fig. 1　System Structure

データベースの中の各節は，導出部から必要に応じて呼び出され，作業記憶に複写される．また導出された Prime Implicant節はデータベースの中に記録される．

④導出部 (Reasoning Module)

双形リテラルについて，導出可能な節の組み合わせ数を計算し，順序づけを行なって Prime Implicant節を導出するとともに，管理部より指示を受け，Prime Implicant節との包含関係から Minimal Support節の高速生成を行なう．

4．2 動作

本研究においては，システムは SONY NEWS-841 上にインタプリタ形式のK-Prologを用いて構築した．システムの起動は問題解決器の中で行なう．データの表現形式とその例を Fig. 2 に示す．

問題解決器においては，節は公理節集合名と節のリストとを項とする複合項の形で表現される（①）．システムは，データ変換部を起動させ，問題解決器のデータベースから公理節集合を取り込み，公理節集合名，節のリスト，クラス，および節のリスト長を項とする複合項の形に変換し，システム内のデータベースに記録する．また導出部を起動させ，この公理節集合について Prime Implicant節を導出し，データベースに記録する（②）．Minimal Support節を生成する時には，問題解決器は，質問節を複合項の形でシステムに対して送り込む（③）．システムは Minimal Support節を生成し（④），問題解決器に戻す．

```
① Expression of a Clause in Reasoner
    pro(group, clause_list)
        ex.   pro(g1, [a, ～b, c])
              pro(g1, [a, b, c])

② Expression of a Clause in the Database of Compiled Hypothesis-based
  Reasoning System
    pro_list(group, clause_list, class, list_length)
        ex.   pro_list(g1, [a, ～b, c], original, 3)
              pro_list(g1, [a, b, c], original, 3)
              pro_list(g1, [a, c], prime_i, 2)
                  original ・・Class Name for Clause from Reasoner
                  prime_i  ・・Class Name for Prime Implicant Clause

③ Expression of a Query Clause from Reasoner
    cms(group, query, Minimal_Supports)
        ex.   cms(g1, ms([a]), MS)

④ Expression of a Minimal Support Clause
    pro_list(group, clause_list, class, list_length)
        ex.   pro_list(g1, [～a], min_sup, 1)
              pro_list(g1, [c], min_sup, 1)
                  min_sup ・・Class Name for Minimal Support Clause
```

Fig. 2 Expression of Clauses

5．評価および考察

今回開発した順序づけ節コンセンサス法の評価を，次の３種類の方法を比較することにより行なった．またコンパイル形式による Minimal Support 節生成の高速化の程度を評価するため，インタプリタ形式との比較を行なった．

①幅優先導出方法
(Breadth-first Method)
②非順序づけ節コンセンサス法
(Non-ordered Clause-Consensus Method)
③順序づけ節コンセンサス法
(Ordered Clause-Consensus Method)

評価に用いた公理集合としては，過去にプロダクションルールによりエキスパートシステムを構築した[17]，船舶の衝突予防に関する，実際の知識集合の部分と同じ内容をもつ命題論理節集合，および乱数により発生させた命題論理節集合とを用いた．前者の知識集合の節の数は最大６０，この時のリテラル数は３６，一つの節中の平均リテラル数は２．８であった．乱数により節を生成する場合，これを参考として一つの節中の平均リテラル数を３とし，また４，８，１６のリテラルについて正負リテラルを発生させ，それぞれのケースについて１６の節を生成した．Table１，Table２において，ＣＰＵ時間は，各ケースとも，あらかじめ設定したそれぞれ３つの質問節に対する時間を平均したものである．なお，時間計測はＣＰＵ時間５０，０００秒で打ち切り，あきらかにこれを超えると思われるものについては計測を省略した．

Prime Implicant節の導出時間（Table１）

幅優先法と比較した場合，節コンセンサス法を用いることにより約５～９０倍以上高速で Prime Implicant節を導出することができた．

ごく小規模ではあるが実際の知識集合を対象と

	Number of Clauses	Number of Literals	CPU Time (Unit : sec)		
			Ordered Consensus Method	Non-ordered Consensus Method	Breadth -first Method
Propositions Generated at Random	16	4	6.0	5.5	97.0
	16	8	66.3	67.5	3019.2
	16	16	477.0	520.9	4864.0
Propositions of a Knowledge Set	16	18	7.6	22.7	33.9
	31	25	53.3	109.8	210.1
	31	26	36.2	443.4	1476.9
	60	36	171.1	978.6	15626.6

Table１ Comparison of the 3 Types of the Resolution for Prime Implicant Clauses

したケースについては，順序づけをしない場合と比較して，順序づけを行なった場合の方がケースによっては１０倍以上の高速化を図ることができた．これは節集合中の双形リテラルの組み合わせ数が平均していないことにより，順序づけの効果が大きく現われたものである．両者の比が約２程度の場合もあるが，これはたまたま，良い順番で導出が行なわれたためであると思われる．順序づけをリテラル毎に行なう方法も実行してみたが，最初に一度だけ行なう方法との差は，今回検討した範囲ではほとんどなかった．

公理節集合を乱数により発生させた場合が，最も効果が無い場合であり，当然のことながら両者にはほとんど差は無かった．実際の知識集合を対象としたケースの方が，節数，リテラル数は多いのにもかかわらず，乱数を用いて生成したケースよりも，節数の増加に伴うＣＰＵ時間の増加は顕著ではなく，実際の知識集合を対象としたケースの方が，順序づけ節コンセンサス法を用いることにより，より高速化を期待できる結果となった．

Minimal Support節の生成時間（Table２）

順序づけ節コンセンサス法を用い，コンパイル形式とインタプリタ形式との比較を行なった．空節が生成されない場合，質問節がユニット節のケースでは，コンパイル形式の方が１０～１００倍程度高速であった．また，質問節が２つのリテラ

	Number of Clauses	Number of Literals	Literals in Query Clause	Minimal Support Clauses	CPU time (Unit:sec)		
					Ordered Consensus Method		Breadth -first Method
					Compiled	Interpreted	
Propositions Generated at Random	16	4	1	un-empty	0.4	7.5	102.4
	16	8	1	un-empty	1.2	118.1	above 50000
			2	un-empty	6.9	146.5	above 50000
			3	empty	4.3	92.4	26.9
	16	16	1	un-empty	9.0	822.0	—
			2	un-empty	121.3	1122.7	—
			3	empty	55.1	120.4	17.9
Propositions of a Knowledge Set	16	18	1	un-empty	1.1	10.3	65.0
			2	empty	1.2	9.7	39.8
	31	25	1	un-empty	1.5	72.3	715.5
			2	empty	3.3	70.2	118.1
	31	26	1	un-empty	1.8	53.6	2260.7
			2	empty	11.7	20.9	79.1
	60	36	1	un-empty	2.6	248.4	above 50000
			2	empty	24.0	172.0	311.6

Table２　Comparison of the 3 Types of the Approach for the Generation of Minimal Support Clauses

ルから構成されるケースでも，１０～２０倍程度の高速化を達成することができた．Minimal Support節を生成する時間と，Prime Implicant節を導出した時間とを合計しても，コンパイル形式の方が２倍以内ではあるが，インタプリタ形式よりも速い結果となっている．空節が生成される場合には，空節が生成される時点に大きく依存しており，全解探索をしないため差が小さくなっているが，２～２０倍程度高速である．Minimal Support節の生成に関しても，実際の知識集合を対象としたケースの方が，順序づけ節コンセンサス法の効果をより期待できる結果となった．

インタプリタ形式による幅優先法と，コンパイル形式とを比較した場合，空節が生成されない場合，コンパイル形式の方が約６０～１００００倍以上高速で Minimal Support節を生成することができた．しかしながら空節が生成される場合には，必ずしも節コンセンサス法の方が速いとはいえなかった．これはコンパイル形式でも，Minimal Support節を生成するためには，リテラルの照合，集合演算および包含関係の検討を行なう必要があり，今回のシステムではこの計算に時間を要したためである．１６のリテラルを用いて，乱数により１６の命題論理節を発生させたケースにおいて，１４１の Prime Implicant節が生成されたように，この Prime Implicant節の数の増加に伴い集合演算等の時間が大きくなる．また，空節が生成されない場合の計算量を少なくすることを第一に考え，導出順序を決めたため，質問節に含まれるリテラルについて先に導出を行なった方が空節が速く生成される場合に，幅優先法より遅くなったものと思われる．

６．おわりに

知識ベースの高速コンパイルを可能とする，Prime Implicant節の高速導出方法の開発，およびコンパイルした知識を利用した Minimal Support節の高速生成法を実現することができた．このコンパイル技術を応用して命題論理を対象とした仮説推論システムを構築した結果，Minimal Support節として空節が生成されない場合には，コンパイルしない場合と比較しても，また単純な幅優先戦略による推論と比較しても，推論の大幅な高速化を図ることができた．この傾向は特に質問節がユニット節の場合に顕著であった．

このシステムにおいては，節中のリテラルの照合や，包含関係の検討が重要であり，この照合等のために時間がかかっていることから，節データの表現形式を照合に便利な形式にすることは，高速化を図るために検討しなければならない技術である．また，今回は双形リテラルだけに注目したが，補リテラルが存在しない単形リテラルを含む二つの節から導出される節は，双形リテラルから導出される節を包含しない場合，単形リテラルの増加に伴い単調に増加し，空節を導出することがない，という単形リテラルの特性を利用した高速化も，検討するに値する技術であると思われる．

公理集合が大きくなった場合には，全ての Prime Implicant節をあらかじめ導出することはコストが大きいため，de KleerがＡＴＭＳとの類似性を指摘した(7)，ＣＳＰのように部分を対象とすることや，導出との共通性が見出されてきている Groebner Baseで行なわれているように(18)，推論の道筋に沿って Prime Implicant節を導出する方法も，高速化のために検討する必要がある．実用的な仮説推論システムを，Reiter他の提案したＣＭＳの枠組を利用して実現するためには，これらの高速化技術を実現することや，対象を述語論理に広げ，仮説推論システムの応用性を高めることが大きな課題である．

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.22-32 (1988).

(2) Doyle, J. : A Truth Maintenance System, Artificial Intelligence 12, pp. 231-272 (1979).
(3) de Kleer, J. : An Assumption-based TMS, Artificial Intelligence 28, pp. 127-162 (1986).
(4) Reiter, R., de Kleer, J. : Foundations of Assumption-based Truth Maintenance Systems : Preliminary Report, Proc. AAAI-87, pp. 183-188 (1987).
(5) Freuder, E. C., : Synthesizing Constraint Expressions, Communications of the ACM, 21, No. 11, pp. 958-966 (1978).
(6) Mackworth, A. K. : Constraint Satisfaction, Encyclopedia of Artificial Intelligence, John Wiley and Son, pp. 205-211 (1987).
(7) de Kleer, J. : A Comparison of ATMS and CSP Techniques, IJCAI-89, pp. 290-296 (1989).
(8) 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の応用に関する基礎研究, 人工知能学会全国大会（第３回）論文集, pp. 179-182 (1989).
(9) Karnaugh, M. : The Map Method for Synthesis of Combinational Logic Circuits, AIEE Trans. and Commun. Electron., Vol. 72, pp. 593-599 (1953).
(10) Quine, W. V. : A Way to Simplify Truth Functions, Amer. Math. Monthly, 62, pp. 627-631 (1955).
(11) McCluskey, E. J. Jr. : Minimization of Boolean Functions, Bell System Tech. J., 35, No. 6, pp. 1417-1444 (1956).
(12) Slagle, J. R., Chang, C. L. and Lee, R. C. : A New Argorithm for Generating Prime Implicants, IEEE Trans., Computers, C-19, No. 4, pp. 304-310 (1970).
(13) 上林弥彦, 岡田康治, 矢島脩三：節展開法を用いた論理関数の主項の生成, 電子通信学会論文誌, J62-D, No. 2, pp. 89-96 (1979).
(14) Tison, P. : Generalization of Consensus Theory and Application to the Minimization of Boolean Functions, IEEE Trans. Electronic Computers, EC-16, No. 4, pp. 446-456 (1967).
(15) Morreale, E. : Recursive Operators for Prime Implicant and Irredundant Normal Form Determination, IEEE Trans. Computers, C-19, No. 6, pp. 504-509 (1970).
(16) 松村尚志, 稲石正明, 鶴田三郎, 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－システム開発過程における評価について－, 日本航海学会論文集, No. 81, pp. 31-38 (1989).
(17) 鶴田三郎, 松村尚志, 稲石正明, 今津隼馬, 杉崎昭生：船舶運航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).
(18) 坂井公, 相場亮：CAL:制約論理プログラミングの理論と実例, 電子情報通信学会研究資料, SS87-28, pp. 9-17 (1988).