ICOT Technical Report: TR-543

TR-543

ATMSを用いた前向き仮説推論システムにおける効率的な推論方式

太田好彦，井上克巳

April, 1990

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

１．まえがき

仮説推論は，常に正しいかどうかわからない知識の集合を仮説集合とし，この中から常に正しい知識の集合と矛盾しない部分集合を加えて結論を導く推論形態であり，不完全な知識ベースに基づく知識システム構築のために重要な要素技術である[1][2]．しかしながら，導入した仮説集合の無矛盾性チェックを行わなければならないために，推論速度の点で問題がある．

従来，命題論理のホーン節から定義された知識ベースの無矛盾性を効率的に管理する機構としてＡＴＭＳ[3]が提案されている．このＡＴＭＳと問題解決器とを結合して仮説推論システムが実現できる．このようにＡＴＭＳと問題解決器とを結合した推論システムの例としては，文献（４）が挙げられる．これはＡＴＭＳに制約言語の問題解決インタフェースを設けた推論システムである．

本論文では，前向き推論システムとＡＴＭＳとを結合した仮説推論システムにおける効率的な推論方式について述べる．このような仮説推論システムで従来提案されている例として，文献（５）や文献（６）が挙げられる．これらでは，ＡＴＭＳとプロダクション・システムとを結合し，Ｒｅｔｅネットワーク[7]内でパターン・マッチングとラベル計算を融合し仮説推論の高速化を図っている．文献（５）は，独自の仮説ネットワークと調整アルゴリズムとを提案し，問題解決器と無矛盾性管理機構とを完全に融合しているが，課題として，多重コンテキストを扱えるようにすることを挙げている．文献（６）では，その処理方式についての詳細は述べられていない．また，文献（８）や文献（９）のシステムでも，前向き推論アルゴリズムにＲｅｔｅアルゴリズムを採用し，無矛盾性管理をＡＴＭＳによって行っているが，Ｒｅｔｅネットワーク内でラベル計算は行われていない．

本論文で述べる仮説推論方式は，Ｒｅｔｅ－ｌｉｋｅネットワークの２入力ノードにおいて中間的なラベル計算を行って保持しておくことにより，仮説推論速度の向上を実現するものである．従来のシステム例と比較した本方法の特徴は，入力知識ベースをＲｅｉｔｅｒの正規デフォルト理論[10]のサブセットに基づいた論理プログラム（一階述語論理ベース）としていること，ＡＴＭＳ

と前向き推論エンジンをモジュール化してそれぞれ別々にも使用できるようにしていること，競合解消を行わず多重コンテキストで推論を行うこと，更に，ラベルの伝播は元々効率的なＡＴＭＳにまかせていること等が挙げられる．

また，本仮説推論方式を採用して，仮説推論システムＡＰＲＩＣＯＴ／０[11]をＰＳＩ－ＩＩ[12]マシン，言語ＥＳＰ[13]上にインプリメントし，その仮説推論時間を実験的に評価し，有効性を示した．

２章において対象とする知識ベースを定義し，３章においてＡＴＭＳの概要を示し，４章で仮説推論システムＡＰＲＩＣＯＴ／０の構成，５章でＲｃｔｅ－ｌｉｋｅネットワーク，６章で推論方式，７章で評価についてそれぞれ述べる．

２．知識ベースの定義

以下で取り扱う知識ベースは，Ｒｅｉｔｅｒの正規デフォルト理論[10]のサブセットに基づいた論理プログラムであり，次に示すＤとＦとにより定義される．

Ｆは，ホーン節の有限集合である．ホーン節は，式（１）または式（２）で表される．

$$\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta \ . \qquad (1)$$

$$\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \bot \ . \qquad (2)$$

ここに，α_i（ｉ＝１，…，ｎ；ｎ≧０）およびβは一階述語論理のアトム（Ａｔｏｍｉｃ ｆｏｒｍｕｌａ）であり，∧は連言，→は含意，⊥は偽（Ｆａｌｓｉｔｙ）である．また，$\alpha_1 \wedge \cdots \wedge \alpha_n$を前件，$\beta$を後件と呼ぶ（節（２）の後件は偽）．更に，節中の全ての変数は頭部において全称記号で束縛されているものとする．ホーン節（１）をＡＰＲＩＣＯＴ／０では，以下の形式で記述する．

$$\mathrm{ID} :: \alpha_1, \cdots, \alpha_n -> \beta \ . \qquad (3)$$

ここに，ＩＤは節名である．また，前件が空（ｎ＝０）の場合のホーン節をＡＰＲＩＣＯＴ／０では，以下の形

式で記述する．

β ．　　（４）

この場合，βは基礎アトムに限る．また，ホーン節（２）をＡＰＲＩＣＯＴ／０では，以下の形式で記述する．

ＩＤ：：α_1，…，α_n－＞［］．　　（５）

Ｄは，α：β／βのような推論規則で定義される正規デフォルトの集合であり，αはアトムα_i（ｉ＝１，…，ｎ：ｎ≧０）の連言，βはアトムとする．ここで，αをデフォルトの前提，βをデフォルトの結論という．これをＡＰＲＩＣＯＴ／０では，以下の形式で記述する．

ＩＤ：：α_1，…，α_n－＞ａｓｓｕｍｅ（β）．　　（６）

ここに，ＩＤは規則名である．また，ここで前提が空（ｎ＝０）の場合，すなわち：β／βの場合，ＡＰＲＩＣＯＴ／０では，以下の形式で記述する．

ａｓｓｕｍｅ（β）．　　（７）

ここに，βは基礎アトムに限る．

３．ＡＴＭＳ

以下に，ＡＴＭＳ[3]の概要について示す．

①ノード

ＡＴＭＳが管理するデータは，命題論理のアトム，あるいは述語論理の基礎アトムかそれらの連言である．データが基礎アトムの連言となるのは，５章で述べるＲｅｔｅ－ｌｉｋｅネットワークの２入力ノードに対応するノードに限る．データに対応してＡＴＭＳ内部では，以下に示すデータ構造のノードが生成される．

Ｄａｔｕｍ：データ．
Ｌａｂｅｌ：ラベル．
Ｊｕｓｔｉｆｉｃａｔｉｏｎｓ：このデータへの理由付けの前件ノードのポインタの集合の集合．

Ｃｏｎｓｅｑｕｅｎｔｓ：このデータを前件とする理由付けの後件ノードのポインタの集合．
Ｃｏｎｔｒａｄｉｃｔｏｒｙ：ＯＵＴなら１，ＩＮなら０とする．

ノードは，データで一元的に管理されている．以下，ある論理式βとその基礎化代入θとで，その論理式の基礎式を$\beta\theta$と記述する．また，基礎式$\beta\theta$に相当するノードを，単に$<\beta\theta>$と記述する．

② 理由付け

ホーン節とその基礎化代入θとで表された式（８）に基づいて，基礎式$\alpha_i\theta$（$i=1,\cdots,n;n\geq 0$）から基礎式$\beta\theta$が導かれたことをＡＴＭＳに理由付け（９）という形式で与える．

$$(\alpha_1\wedge\cdots\wedge\alpha_n\rightarrow\beta)\theta. \qquad (8)$$

$$<\alpha_1\theta>,\cdots,<\alpha_n\theta>\Rightarrow<\beta\theta>. \qquad (9)$$

ここに，$<\alpha_i\theta>,\cdots,<\alpha_n\theta>$をこの理由付けの前件ノードと呼び，$<\beta\theta>$をこの理由付けの後件ノードと呼ぶ．理由付けに相当するホーン節の集合をＪで表す．

③ 仮定

：$\beta\theta$／$\beta\theta$に相当する理由付けを以下のように表す．

$$<\Gamma\ \beta\theta>\Rightarrow<\beta\theta>.$$

ここに，$<\Gamma\ \beta\theta>$を仮定ノードと呼び，$\Gamma\ \beta\theta$を仮定と呼ぶ．

③ 環境

仮定の集合を環境と呼びＥで表す．

④ ラベル

環境ＥとＪよりある基礎式が導けるような全ての環境Ｅの集合をその基礎式の完全ラベルという．また，ある環境ＥとＪより偽が導けるとき，ＥをＪに対して矛盾する環境であるといい，このＥを単に矛盾環境という．ラベルは，完全ラベルから矛盾環境および他を包含する環境を取り除いた環境の集合である．$<\alpha_i\theta>$（$i=1,\cdots,n;n\geq 0$）を前件ノードとする理由付けが与えられたとき，更新が必要なラベルは以下の手順により計算

できる．

ｉ．後件ノードのラベルをＬ，＜$\alpha_i \theta$＞（ｉ　１，…，ｎ；ｎ≧０）のラベルＬｉとして，Ｌ′を計算する．ここに，環境をビットベクタで表現しておくと，$\cup_i E_i$はビット演算ｏｒの繰り返しによりＬ′が計算できる．

Ｌ′＝｛$\cup_i E_i$，｜$E_i \in L_i$｝．

ii．Ｌ∪Ｌ′から矛盾環境および他を包含する環境を取り除いたＬ″を後件ノードの新しいラベルとする．

iii．もしＬ＝Ｌ″ならば終了し，さもなければivを行う．

iv．もし後件が偽ならばＮｏｇｏｏｄｓの処理を行い，さもなければラベルの伝播を行う．ここにＮｏｇｏｏｄｓの処理は，Ｅ∈Ｌ″なるＥを包含する環境を当該ノード（偽に相当するノード）を除く全てのノードのラベルから削除することである．また，ラベルの伝播は，この後件ノードを前件ノードとする理由付けの後件ノードのラベル計算をｉより行うことである．

Ｊと矛盾しないある１つの環境に対応付けて，その仮定の集合の基で導くことができるデータの集合を１つのコンテキストと呼ぶ．あるデータが属する複数のコンテキストを表す環境の極小集合は，そのデータに相当するノードのラベルである．つまり，ＡＴＭＳは複数のコンテキストを一括管理（多重コンテキスト管理）している．

⑤信念の状態

あるデータのラベルが空のとき，信念の状態はＯＵＴであるといい，少なくとも１つの環境がラベルの要素であれば信念の状態はＩＮであるという．

４．仮説推論システムＡＰＲＩＣＯＴ／０の構成

Ｆｉｇ．１は仮説推論システムＡＰＲＩＣＯＴ／０の構成を示すものであり，知識ベースのコンパイラ(14)，推論エンジンとＡＴＭＳとにより構成されている．

コンパイラは，前記知識ベースを入力し，それに対応するＲｅｔｅ－ｌｉｋｅネットワークを構築し出力する．推論エンジンはこれに基づき，多重コンテキストで前向きに推論を実行する．ＡＴＭＳが多重コンテキスト管理を行うことを利用して，推論エンジンでは規則の適用に関する競合解消を行なう必要がない．

推論エンジンは，ＡＴＭＳからある基礎アトムの信念の状態を受信し，導出原理に基づくパターン・マッチン

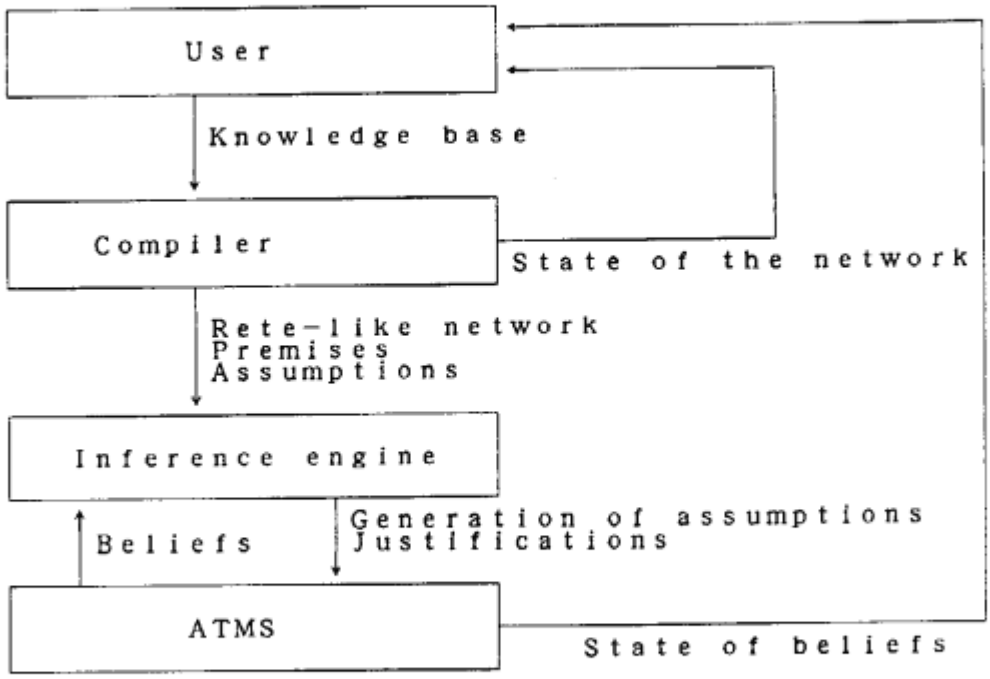


Fig. 1 Configulation of APRICOT/0.

グによりその結論である基礎アトムへの理由付けを生成しＡＴＭＳに送信する．更に，デフォルトの前提の信念の状態がＩＮとなったときにＡＴＭＳに新たな仮定の生成タスクと共に理由付けを送る．ＡＴＭＳは，無矛盾性管理を行い，最終的に推論によって得られた全ての基礎アトムとそのラベルをユーザに表示することができる．

５．Ｒｅｔｅ－ｌｉｋｅネットワーク

ＡＰＲＩＣＯＴ／０の入力となる知識ベースのＦの中で形式（３）（５）の全ての節およびＤの中で形式（６）の全てのデフォルトを対象にＲｅｔｅ－ｌｉｋｅネットワークに展開する．これによって，後で述べる推論実行時において，複数の節の前件やデフォルトの前提にあって共有できるアトムの連言とＩＮの状態である基礎アトムとの間でのユニフィケーション，さらに理由付けに伴うラベル計算の手間を省くことを実現する．従って，形式（４），（７）は，コンパイルの対象外とする．

Ｒｅｔｅ－ｌｉｋｅネットワークは，Ｒｅｔｅネットワークと同様に１個のｒｏｏｔノード，複数の１入力ノード，２入力ノード，終端ノードおよびそれらを結ぶリンクから構成される．Ｒｅｔｅ－ｌｉｋｅネットワークには，基礎式に対応するＡＴＭＳノード（トークンとも呼ぶ．以下，単にノードと書く．実際には，ノードへのポインタでよい．）が流れることにより推論が実行される．

① ｒｏｏｔノード

ｒｏｏｔノードは，スロットＳｕｃｃｅｓｓｏｒを持ち，全ての１入力ノードへのポインタの集合を保持している．

② １入力ノード

以下のスロットを持つ．

Ｄａｔｕｍ：ｆｒｅｅｚｅ（α）．
Ｓｕｃｃｅｓｓｏｒ：２入力ノードへのポインタの集合．
Ｔｅｒｍｉｎａｌ：終端ノード（もし存在すれば）へのポインタの集合．
ＷＭ：推論実行時に使用するワーキング・メモリで，基礎式$\alpha\theta$に対応するノードへのポインタの集合．

ここに，freeze（α）は，論理式αに含まれる全ての変数を新しい定数記号（例えばS，これを変数を表現する定数記号と解釈）に置き換えてできる基礎式を返す（フリーズする）関数とする．また，この逆（メルトする）関数も次のように定義する．

$$melt(freeze(\alpha)) = \alpha .$$

任意の節$\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta$の前件あるいは任意のデフォルト$\alpha_1 \wedge \cdots \wedge \alpha_n : \beta / \beta$の前提に記述されている各アトム$\alpha_i$（$i = 1, \cdots, n$；$n \geq 1$）について，freeze（$\alpha_i$）をDatumとした1入力ノードを生成する．このとき，rootノードのSuccessorスロットにこの1入力ノードへのポインタを追加しておく．この1入力ノードは，Datumについて一元管理されている．

③ 2入力ノード

以下のスロットを持つ．

Datum：freeze（$\alpha \wedge \alpha_j$）．
Successor：2入力ノードへのポインタの集合．
Predecessor：αに対応する2入力ノードおよびα_jに対応する1入力ノードへのポインタの集合．
Terminal：終端ノード（もし存在すれば）へのポインタの集合．
WM：推論実行時に使用するワーキング・メモリで，基礎式（$\alpha \wedge \alpha_j$）θに対応するノードへのポインタの集合．

任意の節$\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta$の前件あるいは任意のデフォルト$\alpha_1 \wedge \cdots \wedge \alpha_n : \beta / \beta$の前提の各アトムの連言$\alpha = \wedge_i \alpha_i$（$i = 1, \cdots, j - 1$；$j \geq 2$）とアトム$\alpha_j$との連言$\alpha \wedge \alpha_j$に対応する2入力ノードが，$j = 2, \cdots, n$について生成される．この2入力ノードは，Datumについて一元管理されている．

④ 終端ノード

節$\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta$に対応する終端ノードを節名IDを用いて#IDで表し，これはノード（10）のようなトークンが流れてきたときに，θがβの基礎化代入となるならば，理由付け（11）をATMSに送る．

$<(\alpha_1 \wedge \cdots \wedge \alpha_n)\ \theta>$．　　（10）

$<(\alpha_1 \wedge \cdots \wedge \alpha_n)\ \theta> \Rightarrow <\beta\ \theta>$．　　（11）

また，デフォルト$\alpha_1 \wedge \cdots \wedge \alpha_n : \beta / \beta$に対する終端ノードを規則名ＩＤを用いて＃ＩＤで表し，これはノード（１０）のようなトークンが流れてきたときに，θがβの基礎化代入となるならば，理由付け（１２）をＡＴＭＳに送る．

$<(\alpha_1 \wedge \cdots \wedge \alpha_n)\ \theta>, <\Gamma\ \beta\ \theta> \Rightarrow <\beta\ \theta>$．
（12）

このような終端ノードは，節の前件またはデフォルトの前提に対応する１入力ノードまたは２入力ノードにおけるＴｅｒｍｉｎａｌスロットからポイントされる．

６．推論方式

仮説推論システムＡＰＲＩＣＯＴ／０の推論部は，前向き推論エンジンとＡＴＭＳとから構成されており，推論エンジンでデータの理由付けを生成してＡＴＭＳに送り，ＡＴＭＳではデータの信念の状態を返すという結合である（Ｆｉｇ．１参照）．

推論エンジンは，１つの待ち行列（初期状態は空）を有している．

まず，Ｒｅｔｅ－ｌｉｋｅネットワークへのコンパイルから除外されていた入力知識ベースのうち，$\beta\theta$を基礎アトムとして，形式（４）に対応する節$\beta\theta$に対して理由付け（１３）を，形式（７）に対応するデフォルト：$\beta\theta / \beta\theta$について理由付け（１４）をＡＴＭＳに送り，待ち行列の末尾に全ての$<\beta\ \theta>$を追加する．

$\Rightarrow <\beta\ \theta>$．　　（13）

$<\Gamma\ \beta\ \theta> \Rightarrow <\beta\ \theta>$．　　（14）

この初期設定の後，Ｆｉｇ．２にＥＳＰ[13]で記述したアルゴリズムに従って推論エンジンは動作する．以下，

```
goal (Queue, Root, ATMS) :-
      :get (Queue, Token),
      :datum (Token, At),
      :successor (Root, Node),
      :datum (Node, D),
      :melt (D, A),
      A=At,
      :add_wm (Node, Token),
      next (Token, Node, ATMS, Queue) ;
goal (Queue, Root, ATMS) ;

next (Token, Node, ATMS, Queue) :-
      :successor (Node, Node_2),
      :predecessor (Node_2, Node_1),
      :wm (Node_1, WME),
      :contradictory (WME, 0),
      :datum (WME, A),
      :datum (Token, F),
      :justification (ATMS, [Token, WME], F^A, Na),
      :add_wm (Node_2, Na),
      next (Na, Node_2, ATMS, Queue) ;
next (Token, Node, ATMS, Queue) :-
      :terminal (Node, ID),
      :execute (ID, Token, ATMS, Na),
      :add (Queue, Na),
      fail ;
```

Fig. 2 Reasoning algorithm on APRICOT/0.

図中の変数記号を括弧内に示し，これについて説明する．

① 正リテラルがｇｏａｌ／３の第１節

推論エンジンは，待ち行列（Ｑｕｅｕｅ）よりＩＮ状態であるノードを先頭から取出し，Ｒｅｔｅ－ｌｉｋｅネットワークのｒｏｏｔノード（Ｒｏｏｔ）から流す．このノードをトークン（Ｔｏｋｅｎ）と呼ぶ．ｒｏｏｔノードとＳｕｃｃｅｓｓｏｒリンクで結合されている１入力ノード（Ｎｏｄｅ）にそのトークンを流す．１入力ノードのＤａｔｕｍをメルトしそれ（Ａ）とトークンのデータ（Ａｔ）がユニファイすれば，その１入力ノードのＷＭにそのトークンを保持させる．

② 正リテラルがｎｅｘｔ／４の第１節

その１入力ノード（Ｎｏｄｅ）とＳｕｃｃｅｓｓｏｒリンクで結合されている２入力ノード（Ｎｏｄｅ＿２）にそのトークンを流す．２入力ノードにおいて，それとＰｒｅｄｅｃｅｓｓｏｒリンクで結合されている他の１入力ノードまたは２入力ノード（Ｎｏｄｅ＿１）のＷＭに保持されているＩＮの状態（信念の状態をチェックするメソッド（１５）による）であるノード（ＷＭＥ）のデータ（Ａ）とトークンのデータ（Ｆ）との連言（Ｆ＾Ａ）をとり，中間的に理由付けをＡＴＭＳに送る．その連言に対応するノード（Ｎａ）を新しいトークンとする．図において，以下のメソッド（１６）は，理由付け（１７）を意味する．

：ｃｏｎｔｒａｄｉｃｔｏｒｙ（ＷＭＥ，０）．　（１５）

：ｊｕｓｔｉｆｉｃａｔｉｏｎ（ＡＴＭＳ，
［Ｔｏｋｅｎ，ＷＭＥ］，Ｆ＾Ａ，Ｎａ）．　（１６）

＜Ｆ＞，＜Ａ＞⇒＜Ｆ＾Ａ＞．　（１７）

この新しいトークンをその２入力ノードのＷＭに保持させ，その２入力ノードとＳｕｃｃｅｓｓｏｒリンクで結合されている他の２入力ノードに流す（ｎｅｘｔ／４の第１節ループ）．以下，同様な操作を行う．

③ 正リテラルがｎｅｘｔ／４の第２節

②の後，Ｓｕｃｃｅｓｓｏｒリンクで結合されている２入力ノードがなくなったときに，そのノード（Ｎｏｄｅ）とＴｅｒｍｉｎａｌリンクで結合されてい

る終端ノード（ＩＤ）を実行する（５章参照）．その結果新しい基礎アトムが生成されれば，それに対応するノード（Ｎａ）を待ち行列に追加する．新しい基礎アトムが生成されず，新しい理由付け（⊥を含む）が生成された場合には，ＡＴＭＳにその理由付けを送るのみでよい．

④正リテラルがｎｅｘｔ／４の第２節でｆａｉｌした後

③の後，オールタナティブとして残っている終端ノード，オールタナティブとして残っているＷＭに保持されているノード，オールタナティブとして残っているＳｕｃｃｅｓｓｏｒリンクで結合されている２入力ノード，オールタナティブとして残っている１入力ノードについても同様な処理を行う．

オールタナティブとして残っているものがなくなった場合に，新しいトークンを待ち行列から取り出し，①より同様に行う．このような操作を繰り返しトークンの待ち行列が空になった場合に，推論エンジンは停止する（正リテラルがｇｏａｌ／３の第２節）．

７．評価

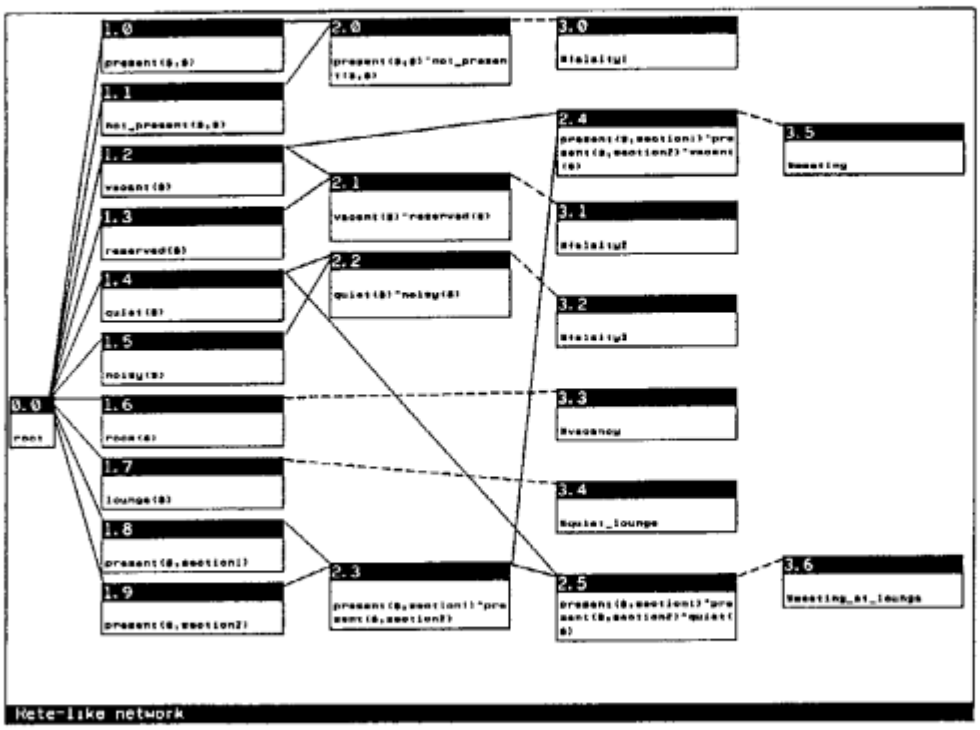
７・１　例題

本推論方式の評価を行うために，次の例題を考える．部門１の１人と部門２の１人と同時にミーティングをしたい．部門１のａとｂ，部門２のｃとｄはこのミーティングに出席してくれると考えられる．また，そのミーティングの場所の候補は，会議室２１２，会議室２１３，会議室２１４，海側ラウンジ，山側ラウンジである．会議室は空いていること，ラウンジは静かであるという条件がある．普通は，会議室は空いていて，ラウンジは静かであると考えてよい．しかし，２１２会議室が予約済みであること，および山側ラウンジは騒々しいことがこのとき既にわかっているものとする．出席者と開催場所を決定せよ．

Ｆｉｇ．３はこの例題を解くための知識ベースの例であり，これに対応するＲｅｔｅ－ｌｉｋｅネットワークをＦｉｇ．４に示す．Ｆｉｇ．４において，Ｘ．Ｙは Ｒｅｔｅノードの識別子であり，Ｘが０ならばｒｏｏｔノード，Ｘが１ならば１入力ノード，２ならば２入力ノード，３ならば終端ノードを示し，Ｙは同じ種類のノードを区別するための番号である．更に，実線は

```
falsity1::
      present(Name, Section),
      not_present(Name, Section)
->
      [].

falsity2::
      vacant(No),
      reserved(No)
->
      [].

falsity3::
      quiet(Side),
      noisy(Side)
->
      [].

vacancy::
      room(No)
->
      assume(vacant(No)).

quiet_lounge::
      lounge(Side)
->
      assume(quiet(Side)).

meeting::
      present(Name1, section1),
      present(Name2, section2),
      vacant(No)
->
      meeting(Name1, Name2, No).

meeting_at_lounge::
      present(Name1, section1),
      present(Name2, section2),
      quiet(Side)
->
      meeting(Name1, Name2, Side).

assume(present(a, section1)).
assume(present(b, section1)).
assume(present(c, section2)).
assume(present(d, section2)).
room(212).
room(213).
room(214).
lounge(seaside).
lounge(mountainside).
reserved(212).
noisy(mountainside).
```

Fig. 3 Knowledge base of the meeting plan.



X, Y: Identifier of a Rete node.

X: 0; Root node, 1; one-input node, 2; two-input node, 3; Terminal node.

Y: Number.

——: Successor (Predecessor) link.

---: Terminal link.

Fig. 4 Rete-like network of the knowledge base shown in the figure 3.

ＳｕｃｃｅｓｓｏｒおよびＰｒｅｄｅｃｅｓｓｏｒリンク，点線はＴｅｒｍｉｎａｌリンクを示すものである．

７・２　比較システム

ＡＰＲＩＣＯＴ／０の推論速度評価のため比較システムを設定した．これは，ＡＴＭＳとＰｒｏｌｏｇ上の慣例的な前向き推論システム（ＣＩＥ：Ｃｏｎｖｅｎｔｉｏｎａｌ　Ｉｎｆｅｒｅｎｃｅ　Ｅｎｇｉｎｅ）とを単純に結合した仮説推論システムであり，中間的なラベル計算は行わずかつ保持していない．Ｆｉｇ．５にＥＳＰで比較システムの推論アルゴリズムを示す．Ｆ，Ｄに含まれる全ての知識は図のようにＥＳＰの節（正リテラルがｒｕｌｅ／２の節）にコンパイルされる．前向き推論は，ｇｏａｌ／２なる述語呼び出しにより行われる．正リテラルがｒｕｌｅ／２の節および正リテラルがｇｏａｌ／２の節の中のメソッドの意味は以下のとおりである．

：ｓｅｔ＿ｆｌａｇ（ＣＩＥ，０）．

ＣＩＥのフラグを０とする．このフラグは，初期設定を１とする．既に，０の場合ｆａｉｌする．このフラグは，ｇｏａｌループで新しい基礎アトムが生成されると１となる．

：ｓｅｔ＿ｆｌａｇ（ＣＩＥ，１）．

ＣＩＥのフラグが１ならば何もしない．０ならば１とする．

：ｉｎ（ＡＴＭＳ，α_i，＜$\alpha_i\theta_i$＞）．

α_iとその基礎化代入θ_iとで，基礎アトム$\alpha_i\theta_i$なるデータをもつＩＮ状態のノードが存在すれば，そのノードを返す（オールタナティブがある）．存在しなければｆａｉｌする．

：ａｓｓｕｍｐｔｉｏｎ（ＡＴＭＳ，$\beta\theta$，
＜$\Gamma\ \beta\ \theta$＞）．

βとその基礎化代入θとで，基礎アトム$\beta\theta$の仮定ノード＜$\Gamma\ \beta\ \theta$＞を生成する．

７・３　本推論方式の効果

```
goal (CIE, ATMS) :-
     :set_flag (CIE, 0),
     rule (CIE, ATMS),
     goal (CIE, ATMS) ;
goal (CIE, ATMS) ;

%% α1∧・・・∧αn→β∈F,
rule (CIE, ATMS) :-
     :in (ATMS, α1, <α1θ1>),
              .
              .
              .
     :in (ATMS, αiθ1・・・θi-1, <αiθ1・・・θi>),
              .
              .
              .
     :in (ATMS, αnθ1・・・θn-1, <αnθ1・・・θn>),
     :justification (ATMS,
       [<α1θ1>, ・・・, <αiθ1・・・θi>, ・・・, <αnθ1・・・θn>],
      βθ1・・・θn, <βθ1・・・θn>),
     :set_flag (CIE, 1),
     fail ;

%% α1∧・・・∧αn:β／β∈D,
rule (CIE, ATMS) :-
     :in (ATMS, α1, <α1θ1>),
              .
              .
              .
     :in (ATMS, αiθ1・・・θi-1, <αiθ1・・・θi>),
              .
              .
              .
     :in (ATMS, αnθ1・・・θn-1, <αnθ1・・・θn>),
     :assumption (ATMS, βθ1・・・θn, <Γβθ1・・・θn>),
     :justification (ATMS,
       [<α1θ1>, ・・・, <αnθ1・・・θn>, <Γβθ1・・・θn>],
      βθ1・・・θn, <βθ1・・・θn>),
     :set_flag (CIE, 1),
     fail ;
       .
       .
       .
rule (CIE, ATMS) ;
```

Fig. 5 Reasoning algorithm based on the conventional inference engine with the ATMS.

Ｔａｂｌｅ１は，Ｒｅｔｅ－ｌｉｋｅネットワークの部分的なノードの動作時におけるＷＭの要素（ＷＭＥ）を示すものである．この表で，述語記号および定数記号は略記してあり，ラベル（Ｌ）はビット・ベクタを十進表示している．これを用いて本方式のラベル計算コストについて比較システムとの対比で考察する．この考察において，前記ラベル計算のうちビット演算ｏｒの回数に着目する．

［ＡＰＲＩＣＯＴ／０］

①　前提が空のデフォルトに対応するノードがトークンとなり，１入力ノード１．８および１．９にトークンが流れてきて，２入力ノード２．３において中間的な理由付けをＡＴＭＳに送り，中間的なラベル計算を行う．このとき，４回ｏｒが実行される．

②　前件が空の節により推論規則ｖａｃａｎｃｙおよびｑｕｉｅｔ＿ｌｏｕｎｇｅが適用されて仮定の生成および理由付けがＡＴＭＳに送られるが，このときはｏｒの回数は０である．

③　②により生成されたノードのうち以下のノードは，それぞれ節ｆａｌｓｉｔｙ２およびｆａｌｓｉｔｙ３により信念の状態はＯＵＴになる．

＜ｖａｃａｎｔ（２１２）＞．
＜ｑｕｉｅｔ（ｍｏｕｎｔａｉｎｓｉｄｅ）＞．

よって，２入力ノード２．４および２．５にはトークンは流れて行かない．２入力ノード２．４および２．５にＩ入力ノードから流れてくるトークンは，それぞれ２個と１個である．これらのトークンのデータにより，２入力ノード２．４および２．５それぞれにおいて，２入力ノード２．３に保持されているノードのデータと連言をとったデータに対する理由付けがＡＴＭＳに送られる．このとき，それぞれ，８回，４回のｏｒが実行される．

④　③で生成された中間的なノードが新しいトークンとなり，それぞれ終端ノード３．５および３．６に送られ，ここではｏｒ実行されず導出データｍｅｅｔｉｎｇ／３の基礎アトムに対応するノードが生成される．

⑤　以上のビット演算ｏｒの回数の総計は，１６回である．

［比較システム］

Table 1 Working memory elements of some nodes on the Rete-like network.

N: Identifier of the Rete node.

WME: Working memory elements of the Rete node.

L: Label of the ATMS node.
Bit-vectors are expressed in decimal notation.

p: present.

s: section.

v: vacant.

q: quiet.

ss: seaside.

ms: mountainside.

^: conjunction.

N	Datum of WME	L
1. 8	p (a, s 1) p (b, s 1)	1 2
1. 9	p (c, s 2) p (d, s 2)	4 8
2. 3	p (a, s 1) ^ p (c, s 2) p (b, s 1) ^ p (c, s 2) p (a, s 1) ^ p (d, s 2) p (b, s 1) ^ p (d, s 2)	5 6 9 10
1. 2	v (2 1 2) v (2 1 3) v (2 1 4)	 32 64
1. 4	q (s s) q (ms)	128
2. 4	(p (a, s 1) ^ p (c, s 2)) ^ v (2 1 3) (p (b, s 1) ^ p (c, s 2)) ^ v (2 1 3) (p (a, s 1) ^ p (d, s 2)) ^ v (2 1 3) (p (b, s 1) ^ p (d, s 2)) ^ v (2 1 3) (p (a, s 1) ^ p (c, s 2)) ^ v (2 1 4) (p (b, s 1) ^ p (c, s 2)) ^ v (2 1 4) (p (a, s 1) ^ p (d, s 2)) ^ v (2 1 4) (p (b, s 1) ^ p (d, s 2)) ^ v (2 1 4)	37 38 41 42 69 70 73 74
2. 5	(p (a, s 1) ^ p (c, s 2)) ^ q (s s) (p (b, s 1) ^ p (c, s 2)) ^ q (s s) (p (a, s 1) ^ p (d, s 2)) ^ q (s s) (p (b, s 1) ^ p (d, s 2)) ^ q (s s)	133 134 137 138

① 節ｍｅｅｔｉｎｇでは，前件に書かれているアトムが３個あり，それぞれとユニファイ可能な基礎アトムが２個づつ存在し，それぞれ１個の環境を有しているため，１６回ｏｒ（８通りの組み合わせについて２回のｏｒがとられる）が実行される．

② 節ｍｅｅｔｉｎｇ＿ａｔ＿ｌｏｕｎｇｅでは，前件にアトムｐｒｅｓｅｎｔ／２が２個書かれており，それぞれにユニファイ可能な基礎アトムが２個づつ存在し，更にアトムｑｕｉｅｔ／１とユニファイ可能な基礎アトムが１個存在している．これらの基礎アトムが，それぞれ１個の環境を有しているので，８回ｏｒ（４通りの組み合わせについて２回のｏｒがとられる）が実行される．

③ 以上のビット演算ｏｒの回数の総計は，２４回である．

よって，本方法の方が８回ビット演算ｏｒが削減される．これに付随してラベル計算ii の処理も削減される．

より一般に，本方法の効果について述べる．節（１８）がＦの要素であるとする．また，θを論理式αの任意の基礎化代入とし，基礎式$\alpha\theta$に相当するノードの環境数を全てのθについて総和をとった数を論理式αの環境数と呼び，Ｎ（α）で表すこととする．$<\beta\theta>$のラベルを求めるための比較システム上で必要なビット演算ｏｒの回数をＮｃ，ＡＰＲＩＣＯＴ／０上で必要なビット演算ｏｒの回数をＮａとすると，Ｎｃ，Ｎａはそれぞれ式（１９），式（２０）に示すようになる．

$$\alpha_1, \cdots, \alpha_{n-1}, \alpha_n \rightarrow \beta . \qquad (18)$$

$$Nc = (n-1)\prod_{i=1}^{n} N(\alpha_i) . \qquad (19)$$

$$Na = \sum_{i=2}^{n} \{ N(\wedge_{j=1}^{i-1} \alpha_j) \cdot N(\alpha_i) \} \qquad (20)$$

ここに$1 \leqq j \leqq n$として，

$$N(\wedge_{i=1}^{j} \alpha_i) \leqq \prod_{i=1}^{j} N(\alpha_i) \qquad (21)$$

である．不等号がついているのは，左辺は右辺から矛盾環境を取り除いた数になるからである．いま，等号の場

合についてＮａとＮｃとの比をとると式（２２）のようになる．これは，$\Pi_i N(\alpha_i)$ 個の環境のうち，矛盾環境がないと仮定したものであり，矛盾環境が生成される場合には本方法による効果は実際にはより大きい．

$$Na/Nc = [1/\{N(\alpha_3)\cdot N(\alpha_4)\cdot\cdots\cdot N(\alpha_n)\} + 1/\{N(\alpha_4)\cdot N(\alpha_5)\cdot\cdots\cdot N(\alpha_n)\} + \cdots + 1/N(\alpha_n) + 1]/(n-1). \quad (22)$$

式（２２）によって，ＡＰＲＩＣＯＴ／０上でのビット演算ｏｒの回数が比較システム上でのビット演算ｏｒの回数よりも小さくなるのは，ｉ＝３，４，…，ｎの任意のｉについて，$N(\alpha_i) \geqq 2$ であるときである（全てのｉについて$N(\alpha_i)=1$のときは比は１）．つまり，２入力ノードがありかつその２入力ノードのＳｕｃｃｅｓｓｏｒリンクで指示される２入力ノードに１入力ノードから複数のトークンまたは複数の環境をラベルに持ったトークンが流れ込む場合ビット演算ｏｒの回数が減少すると結論付けられる．

また，γをアトムとして，節（２１）がＦの要素であり，既に節（１８）に対する$<\beta\ \theta>$のラベルが計算された後ならば，このときの$<\beta\ \theta>$のラベル（節（１８）と節（２３）の後件が同じである必要はない．）を求めるための比較システム上で必要なビット演算ｏｒの回数Ｎｃ′，ＡＰＲＩＣＯＴ／０上で必要なビット演算ｏｒの回数Ｎａ′はそれぞれ式（２４），式（２５）に示すようになる．

$$\alpha_1, \cdots, \alpha_{n-1}, \gamma \rightarrow \beta. \quad (23)$$

$$Nc' = (n-1)\cdot\{\prod_{i=1}^{n-1} W(\alpha_i)\}\cdot W(\gamma). \quad (24)$$

$$Na' = W(\bigwedge_{j=1}^{n-1} \alpha_j)\cdot W(\gamma). \quad (25)$$

この場合も（２１）式で両者が等しいとして，Ｎａ′とＮｃ′との比をとると式（２６）のようになる．

$$Na'/Nc' = 1/(n-1). \quad (26)$$

式（２６）によって，ＡＰＲＩＣＯＴ／０上でのビット演算ｏｒの回数が比較システム上でのビット演算ｏｒの回数よりも小さくなるのは，ｎが２を越える場合である．つまり，２入力ノードがありかつその２入力ノードが，前件のアトムの数が３以上の複数の節に共通である場合，ビット演算ｏｒの回数が減少すると結論付けられる．

これらの考察は，デフォルトについても同様である．

以上より，知識ベースをＲｅｔｅ－ｌｉｋｅネットワークにコンパイルしたとき，２入力ノードがありかつその２入力ノードのＳｕｃｃｅｓｓｏｒリンクで指示される２入力ノードに１入力ノードから複数のトークンまたは複数の環境をラベルに含むトークンが流れ込む場合，またはその２入力ノードが前件のアトムの数が３以上の複数の節または前提のアトムの数が３以上の複数のデフォルトに共通である場合に，ビット演算ｏｒの回数が減少し，本方法による高速化の効果が有効となる．

７・４　推論時間に関する評価実験

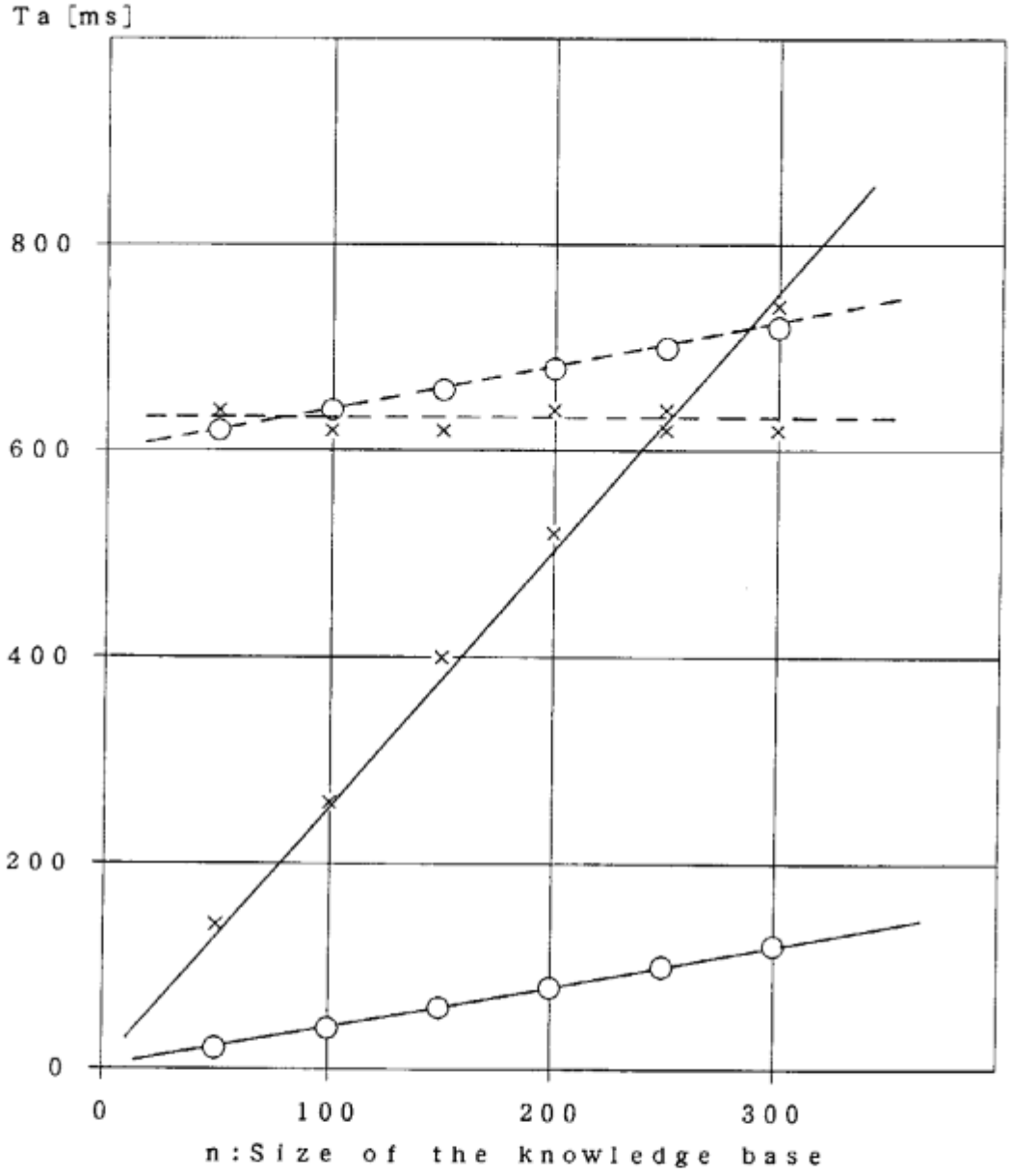
Ｆｉｇ．６は知識ベース規模に応じた本方法の効果を評価するためのテスト知識ベースである．会議の参加者を１０名，空き会議室の数をｎとして，ｎをパラメータとした実験を行う．

前述の結果を用いて，このテスト知識ベースにおいて，ビット演算ｏｒの総回数比は（ｎ＋９）／（１０・ｎ）である．よって，ｎが十分大きい場合では，ＡＰＲＩＣＯＴ／０は比較システムに比べビット演算ｏｒの回数が１／１０になる．これは，ビット演算ｏｒの回数にのみ着目した結果であり，他の処理時間も含めた実験結果はＦｉｇ．７に示されている．図より，ＡＰＲＩＣＯＴ／０は比較システムに比べ，知識ベースの規模によらず，約７倍推論速度が速いことが明らかとなった．

また，ＡＰＲＩＣＯＴ／０上で入力知識ベースからＲｅｔｅ－ｌｉｋｅネットワークにコンパイルする時間は，知識ベースの規模に応じて増加している．一方，比較システムにおいて入力知識ベースからＥＳＰプログラムにコンパイルする時間は，知識ベースの規模によらずほぼ一定である．しかし，コンパイル時間と推論時間との和で考察すると，実験の範囲内において，知識ベース

```
assume(present(name1)).
assume(present(name2)).
      .
      .
      .
assume(present(name10)).

assume(vacant(1)).
assume(vacant(2)).
      .
      .
      .
assume(vacant(n)).

      .
      .
      .

IDj::
      present(name1),
      present(name2),
            .
            .
            .
      present(name10),
      vacant(i)
->
      meeting(name1, name2, ···, name10, i).

      .
      .
      .


%%  i=1, 2, ···, n.
%%  Number of clauses=n.
%%  Number of defaults=n+10.
```

Fig. 6 Tested knowledge base.



○———:Reasoning on APRICOT/0.
×———:Reasoning on the CIE with ATMS.
○- - - - -:Compiling on APRICOT/0.
×- - - - -:Compiling on the CIE with ATMS.

Ta=T/n,
where T means the total execution time,
n = Number of clauses,
n+10 = Number of defaults.

Fig. 7 Evaluation of APRICOT/0.

の規模によらずＡＰＲＩＣＯＴ／０の方が比較システムよりも速いとことが結論付けられる．

８．むすび

以上，前向き推論システムとＡＴＭＳとを結合した仮説推論システムにおいて，効率的な推論方式について述べた．

本仮説推論方式は，Ｒｅｔｅ－ｌｉｋｅネットワークの２入力ノードにおいて中間的なラベル計算を行って保持しておくことにより，仮説推論速度の向上を実現している．本方法による高速化の効果は，与えられた知識ベースをＲｅｔｅ－ｌｉｋｅネットワークにコンパイルしたとき，２入力ノードがありかつその２入力ノードのＳｕｃｃｅｓｓｏｒリンクで指示される２入力ノードに１入力ノードから複数のトークンまたは複数の環境をラベルに含むトークンが流れ込む場合，またはその２入力ノードが前件のアトムの数が３以上の節または前提のアトムの数が３以上のデフォルトに共通である場合に有効である．

この方式を用いて，仮説推論システムＡＰＲＩＣＯＴ／０をＰＳＩ－ＩＩ上にインプリメントした．また，同じマシン上に比較システムとして，この推論方式を採用しない仮説推論システムをインプリメントし，推論実行速度について評価を行った．その結果によれば，ＡＰＲＩＣＯＴ／０は比較システムに比べ，知識ベースの規模によらず，単位知識当たりの処理時間について推論速度が定数倍速いことがわかった．

一方，単位知識当たりのコンパイル時間について，ＡＰＲＩＣＯＴ／０は知識ベースの規模に応じて増加，比較システムは知識ベースの規模によらずほぼ一定である．しかし，推論速度まで考慮にいれると全体として，ＡＰＲＩＣＯＴ／０の方が速いといえる．尚，大規模な知識ベース・システムの開発で，開発完了までに何度も知識ベースの修正を繰り返すような場合，知識ベースからＲｅｔｅ－ｌｉｋｅネットワークへのインクリメンタル・コンパイル法[11][14]もまた既に提案されている．

また，仮説推論システムＡＰＲＩＣＯＴ／０上に論理回路合成問題[15]に対する知識ベースをインプリメントし，本方式の推論速度に関する有効性を確認している．

尚，各知識を共有化できないような知識ベースに基づく仮説推論の高速化が課題として挙げられる．このとき，特に並列処理技術の適用を考えて行きたい．

謝　　辞

本研究の機会を与えて下さり，常に御指導頂いている ICOT 渕一博所長，古川康一次長，第五研究室生駒憲治室長に深く感謝致します．ならびに，御指導頂いた現 NTT 藤井裕一前室長に深く感謝致します．また，日頃討論して頂いている第五研究室各研究員の方々に感謝致します．

◇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）Ｉｎｏｕｅ，Ｋ．：Ｐｒｏｂｌｅｍ Ｓｏｌｖｉｎｇ ｗｉｔｈ Ｈｙｐｏｔｈｅｔｉｃａｌ Ｒｅａｓｏｎｉｎｇ，Ｐｒｏｃ．ｏｆ ｔｈｅ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｆｉｆｔｈ Ｇｅｎｅｒａｔｉｏｎ Ｃｏｍｐｕｔｅｒ Ｓｙｓｔｅｍｓ，Ｖｏｌ．３，ｐｐ．１２７５－１２８１（１９８８）．

（３）ｄｅＫｌｅｅｒ，Ｊ．：Ａｎ Ａｓｓｕｍｐｔｉｏｎ－ｂａｓｅｄ ＴＭＳ，Ａｒｔｉｆ．Ｉｎｔｅｌｌ．，Ｖｏｌ．２８，ｐｐ．１２７－１６２（１９８６）．

（４）ｄｅＫｌｅｅｒ，Ｊ．：Ｐｒｏｂｌｅｍ Ｓｏｌｖｉｎｇ ｗｉｔｈ ｔｈｅ ＡＴＭＳ，Ａｒｔｉｆ．Ｉｎｔｅｌｌ．，Ｖｏｌ．２８，ｐｐ．１９７－２２４（１９８６）．

（５）飯島泰裕，成田良一，吉田裕之，泉寛幸：仮説ネットワークを用いた仮説推論器，人工知能学会研究会研究会資料ＳＩＧ－ＦＡＩ－８７０１－２，ｐｐ．７－１４（１９８７）．

（６）飛鳥井正道，森沢秀一，村田真人，浅野俊昭：エキスパート

システム構築ツールＣＨＯＲＵＳ（２）―仮説推論機能―，情報処理学会第３７回全国大会講演論文集，Ｖｏｌ．２，ｐｐ．１２１８－１２１９（１９８８）．
（７）Ｆｏｒｇｙ，Ｃ．Ｌ．：Ｒｅｔｅ：Ａ Ｆａｓｔ Ａｌｇｏｒｉｔｈｍ ｆｏｒ ｔｈｅ Ｍａｎｙ Ｐａｔｔｅｒｎ／Ｍａｎｙ Ｏｂｊｅｃｔ Ｐａｔｔｅｒｎ Ｍａｔｃｈ Ｐｒｏｂｌｅｍ，Ａｒｔｉｆ．Ｉｎｔｅｌｌ．，Ｖｏｌ．１９，ｐｐ．１７－３７（１９８２）．
（８）Ｆｌａｎｎ，Ｎ．Ｓ．，Ｄｉｅｔｔｅｒｉｃｈ，Ｔ．Ｇ．ａｎｄ Ｃｏｒｐｒｏｎ，Ｄ．Ｒ．：Ｆｏｒｗａｒｄ Ｃｈａｉｎｉｎｇ Ｌｏｇｉｃ Ｐｒｏｇｒａｍｍｉｎｇ ｗｉｔｈ ｔｈｅ ＡＴＭＳ，Ｐｒｏｃ．ｏｆ ＡＡＡＩ－８７，ｐｐ．２４－２９（１９８８）．
（９）Ｊｕｎｋｅｒ，Ｕ．：Ｒｅａｓｏｎｉｎｇ ｉｎ Ｍｕｌｔｉｐｌｅ Ｃｏｎｔｅｘｔｓ，Ｗｏｒｋｉｎｇ Ｐａｐｅｒ，Ｎｏ．３３４，ＧＭＤ（１９８８）．
（１０）Ｒｅｉｔｅｒ，Ｒ．：Ａ Ｌｏｇｉｃ ｆｏｒ Ｄｅｆａｕｌｔ Ｒｅａｓｏｎｉｎｇ，Ａｒｔｉｆ．Ｉｎｔｅｌｌ．，Ｖｏｌ．１３，ｐｐ．８１－１３２（１９８０）．
（１１）井上克巳，太田好彦：仮説推論システムＡＰＲＩＣＯＴ／

0による知識コンパイル，人工知能学会研究会資料ＳＩＧ－ＫＢＳ－８８０５－６，ｐｐ．５１－６０（１９８９）．
（１２）Ｎａｋａｓｈｉｍａ，Ｈ．　ａｎｄ　Ｎａｋａｊｉｍａ，Ｋ．：Ｈａｒｄｗａｒｅ　Ａｒｃｈｉｔｅｃｔｕｒｅ　ｏｆ　ｔｈｅ　Ｓｅｑｕｅｎｔｉａｌ　Ｉｎｆｅｒｅｎｃｅ　Ｍａｃｈｉｎｅ：ＰＳＩ－ＩＩ，Ｐｒｏｃ．ｏｆ　ｔｈｅ　Ｓｙｍｐｏｓｉｕｍ　ｏｎ　Ｌｏｇｉｃ　Ｐｒｏｇｒａｍｍｉｎｇ，ｐｐ．１０４－１１３（１９８７）．
（１３）Ｃｈｉｋａｙａｍａ，Ｔ．：Ｕｎｉｑｕｅ　Ｆｅａｔｕｒｅｓ　ｏｆ　ＥＳＰ，Ｐｒｏｃ．ｏｆ　ｔｈｅ　Ｉｎｔｅｒｎａｔｉｏｎａｌ　Ｃｏｎｆｅｒｅｎｃｅ　ｏｎ　Ｆｉｆｔｈ　Ｇｅｎｅｒａｔｉｏｎ　Ｃｏｍｐｕｔｅｒ　Ｓｙｓｔｅｍｓ，ｐｐ．２９２－２９８（１９８４）．
（１４）太田好彦，井上克巳：ＡＴＭＳによるＲｅｔｅ－Ｌｉｋｅネットワークの逐次構築，情報処理学会第３８回全国大会講演論文集，Ｖｏｌ．１，ｐｐ．４２０－４２１（１９８９）．
（１５）Ｍａｒｕｙａｍａ，Ｆ．，Ｋａｋｕｄａ，Ｔ．，Ｍａｓｕｎａｇａ，Ｙ．，Ｍｉｎｏｄａ，Ｙ．，Ｓａｗａｄａ，Ｓ．　ａｎｄ　Ｋａｗａｔｏ，Ｎ．：ｃｏ－ＬＯＤＥＸ：Ａ　Ｃｏｏｐｅｒａｔｉｖｅ　Ｅｘｐｅｒｔ　Ｓｙｓｔｅｍ　ｆｏｒ

Logic Design, Proc. of the International Conference on Fifth Generation Computer Systems, Vol. 3, pp. 1299-1306 (1988).