TR-633

段階的前向き仮説推論システム

太田　好彦、井上　克巳

March, 1991

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

段階的前向き仮説推論システム

An Incremental Hypothesis-based Forward-reasoning System

太田　好彦*　　井上　克己*
Yoshihiko Ohta　　Katsumi Inoue

＊（財）新世代コンピュータ技術開発機構
Institute for New Generation Computer Technology, Tokyo 108, Japan.

Keywords: hypothesis-based reasoning, default reasoning, ATMS, belief revision, Rete algorithm.

Ｓｕｍｍａｒｙ：

Hypothesis-based reasoning is a basic technique for building AI systems based on incomplete knowledge. The hypothesis-based reasoning system derives conclusions despite the absence of total domain knowledge. If we give new knowledge to the system, beliefs of conclusions should be partially revised on the system.

The assumption-based truth maintenance system (ATMS) is a hypothesis-based reasoning system whose inputs are a set of propositional Horn clauses and a set of hypotheses. The ATMS is often used with a problem solver because we need logical variables to represent domain knowledge.

APRICOT/0, a hypothesis-based forward-reasoning system, consists of an compiler, the Rete-based inference engine and the ATMS. The compiler translates first-order Horn clauses and normal defaults into a Rete-like network. A Rete-like network is made up of a root node, one-input nodes, two-input nodes and terminal nodes. Allowing faster hypothesis-based reasoning, the inference engine gives intermediate justifications to the ATMS at two-input nodes and store the sets of hypotheses where those intermediate data hold. However, the whole of the input knowledge base has to be recompiled and the beliefs of conclusions have to be revised on the whole when the input knowledge base is partially modified by the user. Therefore, an incremental compiler of the input knowledge base into the Rete-like network and an incremental inference system where beliefs of conclusions are partially revised with additional knowledge including variables are required.

This paper presents a incremental hypothesis-based reasoning system which includes the incremental compiler and the incremental inference system. The incremental compiler is implemented with a modified ATMS which is called RATMS. The RATMS can efficiently offer the inference engine the updated parts of the Rete-like network. Beliefs of previous conclusions can be partially revised with additional knowledge including variables. The incremental hypothesis-based reasoning system has been implemented, and has been applied to a logic circuit design problem. The expermental result shows that beliefs of solutions are efficiently revised by additional knowledge for datapath design.

1．まえがき

仮説推論は，常に正しい知識である事実の集合と常に正しいとは限らない知識である仮説の集合とに基づき，事実の集合と矛盾しない仮説部分集合を加えて結論を導く推論形態である[1][2]．既に，事実や仮説をインクリメンタルに入力し現時点までに与えられた事実の集合と仮説の集合とにより多重コンテキストで仮説推論を行うAssumption－based Truth Maintenance System(ATMS)[3]が提案されている．ATMSは，後から追加される事実によって，今まで信じられていた結果が信じられなくなるというような信念の翻意を効率的に行う機能をもつ．しかしながら，事実を命題論理のホーン節，仮説を命題論理のアトムに限定しているので，適当な問題解決器とともに用いられる．

例えば，文献(4)は，ATMSに制約言語の問題解決インタフェースを設けたシステムである．また，一階述語論理のホーン節と正規デフォルト[5]とにより表現された知識の集合を扱う前向き仮説推論システムにAPRICOT/0[6][7]がある．これは，ある基底仮説の集合のもとで信じられている基礎アトムと入力知識ベースの要素とのパターン・マッチングにより信んじうる基礎アトムを導く前向き推論エンジンと，基礎アトムの信念の状態を管理するATMSとから構成されている．ATMSは推論エンジンから与えられるデータをそれが成り立つ基底仮説の集合の集まりでラベル付けする．このラベルの計算は，Rete[8]に類似したネットワークの２入力ノードにおいて中間的な理由付けをATMSに送りその結果を保存する方法によって効率化できる[7]．このシステムにおいては，変数を含まない知識が後から知識ベースに追加される場合，ATMSの機能を用いて帰結の信念の状態の更新ができる．

しかしながら，帰納推論等によって生成した知識を用いて問題解決をするような場合や多くの深い知識，常識的知識あるいは例外を含む知識の中から段階的に知識を加えて問題解決をするような場合等においては，変数を含む知識が後から知識ベースに追加される．この場合，再び全ての知識を最初からコンパイルし直し，新たに推論を行うのは非効率である．そこで，対応するReteネットワークを部分的に修正して，Reteネットワークの変更分から推論を始めて帰結の信念の状態の部分的修正を施すような機構が要求される．

従来，プロダクション・システムを対象に，後からルールが追加された場合における，Reteネットワークのインクリメンタル・コンパイラがいくつか提案されている．荒屋ら[9]は，Reteアルゴリズムを利用したReteネットワークの逐次構築法を提案している．この方法は，Reteネットワークの修正段階でReteノード・メモリを初期化しなければならないので，変更後の推論は最初から行わなければならない．Lecら[10]は，一般化されたReteネットワークに動的なルール追加を可能とする方法を提案している．しかしながら，プロダクション・システムには，帰結の集合と入力知識集合との整合性維持機能がないので，最初からあるルールを含めて推論した帰結の集合と後からそのルールを追加して推論した帰結の集合は必ずしも一致しないという問題点がある．

本論文では，前向き仮説推論システムにおける段階的推論のためのインクリメンタル・コンパイラとその出力に基づく段階的な推論方式について述べる．このインクリメンタル・コンパイラは，ATMSに修正を施して実現される[11]．従来のReteネットワークのインクリメンタル・コンパイラと比較すると，Reteノード・メモリを初期化する必要がなく，以前の推論実行の中間結果を用いて効率的に入力知識ベース変更後の推論が行えるという利点がある．さらに，正規デフォルト理論[5]に基づいてシステムの出力が決まるため，最初からある知識を含めて推論した帰結の集合と後からその知識を追加して推論した帰結の集合とが一致する．

なお，既にインプリメントされているAPRICOT/0に，この論文で述べられているインクリメンタル・コンパイラと推論エンジンを組込み，本段階的前向き仮説推論システムの有効性を確認した．

2章では，ATMSについて概説する．3章で，段階的前向き仮説推論システムについて述べる．3・1節でそのシステムの入出力を明示し，3・2節でシステム構成とその動作の概略について示す．3・3節でインクリメンタル・コンパイラの実現方法を，3・4節でその出力を用いて段階的に推論する方式を示す．4章でその段階的前向き仮説推論システムを評価する．5章では，他の研究との関連について述べる．

2．ATMS

ATMS[3]は，命題論理のホーン節および命題論理のアトムである仮説(Assumption)が入力され，各命題（Datum:データとも呼ぶ）を以下に示すデータ構造（これをATMSノードと呼ぶ）でその信念(Belief)の状態を管理するシステムである．

<Datum, Label, Justifications, Consequents>.

P_i ($i=1, \cdots, n; n \geq 0$)およびQを命題論理のアトム，記号“⊥”を偽とする．このとき，ATMSに入力されるホーン節は理由付けと呼ばれ，以下のいずれかの形式で記述される．

$$P_1, \cdots, P_n \Rightarrow Q. \quad (1)$$

$$\Rightarrow Q. \quad (2)$$

$$P_1, \cdots, P_n \Rightarrow \bot. \quad (3)$$

ここに，記号“⇒”の左を理由付けの前件，右を理由付けの後件，理由付けの前件の各アトムに対応するATMSノードを前件ノード，理由付けの後件に対応するATMSノードを後件ノードと呼ぶ．

また，Qが真であるという仮説をQと区別してΓ_Q（対応するATMSノードを仮説ノードと呼ぶ）と記述する．例えば，正規デフォルト：Q／Qは，以下の理由付けに対応する．

$$\Gamma_Q \Rightarrow Q.$$

現時点までに入力されたホーン節の集合をJ，現時点までに入力された仮説の集合をHと記す．Hの部分集合を環境(Environment)と呼びEで表す．Jとある環境から⊥が導かれるときその環境を矛盾環境(Nogood)と呼び，Jとある環境から⊥が導けないときその環境を無矛盾な環境と呼ぶ．ある2つの環境E_i，E_jに$E_i \subseteq E_j$なる関係があるとき，E_iをE_jの下位環境と呼び，E_jをE_iの上位環境と呼ぶ．ある環境の集合Lにおいて，次式で定義される集合μLをLの極小集合と呼ぶ．

$$\mu L = \{E \mid E \in L, \ E\text{と異なるLの任意の要素はEの下位環境でない}\}.$$

任意のデータQに対応するATMSノードのラベルはμL_Qである．ここに，

$$L_Q = \{E \mid E \subseteq H, \ J \cup E \vdash Q, \ E\text{は無矛盾な環境}\}$$

である．

環境は，以下のようなデータ構造（環境ノードと呼ぶ）で管理される．

<Assumptions, Nodes>.

Assumptionsスロットには，後述のラベル計算の高速化のために，ビットベクタが格納される．環境のビットベクタ表現は，環境の要素である仮説に対応するビット位置を1，他を0とするようなビットベクタで表わすことである．ATMSノードのLabelスロットには，そのデータのラベルの要素の環境に対応する環境ノードへのポインタ(Label link)のリストが格納される．また，環境ノードのNodesスロットには，この環境ノードを指示するポインタがLabelの要素になっているATMSノードへのポインタ(Node link)のリストが格納される．

無矛盾な環境に対応する環境ノードは，ビットベクタをキーとするハシュ表（これを

Environment Hash Tableと呼ぶ:ETと略記）で管理される．また，矛盾環境に対応する矛盾環境ノードの集合の極小集合は，Nogood Database(NDと略記)で管理される．

E_iがE_jの下位環境（E_jはE_iの上位環境）であるとき，それぞれに対応するビットベクタをV_i，V_jとする．それぞれに対応する環境ノードをε_{Vi}，ε_{Vj}とすれば，ε_{Vi}はε_{Vj}の下位環境ノード，ε_{Vj}はε_{Vi}の上位環境ノードと呼ばれる．

いま，形式(1)または(2)の理由付けがATMSに与えられたとすると，ラベル計算アルゴリズムは以下のとおりである．

［ステップ１］　Justificationsスロットに前件ノードへのポインタ(Justification link)のリストを追加する．また，各々のP_iに対応するATMSノードのConsequentsスロットにこの後件ノードへのポインタ(Consequent link)を追加する．

［ステップ２］　前件ノードへのポインタのリストをPとして，Fig.1に示す手続きを実行し，L_n' :=Environments($\{\varepsilon_0\}$,P,ET)とする（nは，前件のアトムの数）．ここに，ε_0は空の仮説集合に対応する環境ノードを表す．以下，このように環境ノードに対応するビットベクタを十進法で表した数をεの添字としてその環境ノードを表すこともある．Fig.1で，ORはビット演算ORを示し，Member(V,ET)はある環境に対応するビットベクタをVとして，〈V,_〉なる環境ノードがETに登録されていればそのポインタ（これををε_Vと書く）を返し，登録されていなければnoを返す関数とする．

なお，手続きEnvironmentsを実行することは，各前件P_iに対応するATMSノードのラベルをL_iとして，集合$\{x \mid x = \cup E_i,\ E_i \in L_i\}$を計算することに相当している．

［ステップ３］　以前の後件ノードのLabelスロットの内容をLとして，$L \cup L_n'$ からNDの各要素の上位環境ノードを取り除いた集合，さらにその集合の極小集合をL''とする．このL''を後件ノードのLabelスロットに格納する．L''の各要素の環境ノードのNodesスロットにこの後件ノードへのポインタがなければ追加する．

［ステップ４］　もし$L = L''$ならば終了し，さもなければ以下の処理を実行する．

［ステップ５］　この後件ノードのConsequentsスロットの要素が指示するATMSノードのJustificationsスロットの各要素（リスト）でこの後件ノードへのポインタが存在する理由付けに対応するラベル計算をステップ２より行う．

また，形式(3)の理由付けがATMSに与えられたとする．このときのラベル計算は，ステップ１からステップ４までを同様としステップ５を以下の処理とする．

［ステップ５′］　L''の各要素ε''に対して，次の処理を行う．ε''をNDに登録し，もし$\varepsilon \in$ETなるεがε''の上位環境ノードであるとき，εのNodesスロットの各要素が指示するATMSノードのLabelスロットからεを削除する．その後，εをETから削除する．

ATMSノードの状態には，現時点でそのデータが真であると信じられている（IN状態あるいはBelievedと呼ぶ）あるいは信じられていない（OUT状態と呼ぶ）の２つの状態がある．OUT状態のATMSノードは，Labelスロットが空である．

３．段階的前向き仮説推論システム

３・１　入出力

本システムには，一階述語論理のホーン節(Horn clause)の集合Wtおよび正規デフォルト(Normal default)[5]の集合Dt (t=0,1,…)が段階的に入力される．その出力は，現時点までに入力されたWtの和集合とDtの和集合（$\cup W_t$，$\cup D_t$）のもとで信じられている基礎アトムの集合である．

ホーン節は，以下のいずれかの形式で記述される．

$$ID::A_1 \wedge \cdots \wedge A_n \rightarrow B. \qquad (4)$$

```
procedure Environments (L0', P, ET) :
begin
i:=0;
X:=P;
while X is not empty do
 begin
 i:=i+1;
 let Pi be the first node of the list X, and X:=X- {Pi} ;
 make Li' empty;
 for all ε'i-1 in Li-1' do
  begin
  let Vi-1' be Assumptions of εi-1' ;
  for all εi in Label Li of Pi do
   begin
   let Vi be Assumptions of εi;
   Vi' =OR (Vi 1', Vi) ;
   if Member (Vi', ET) =no then
    begin
    let εi' be <Vi', []>; ..... (*)
    insert εi' in ET
    end;
   else
    εi' :=Member (Vi', ET) ;
   concatenate εi' to Li'
   end
  end
 end;
return Li'
end.
```

Fig. 1 Procedure Environments used in the label computation.

$$B. \tag{5}$$

$$ID::A_1 \wedge \cdots \wedge A_n \rightarrow \perp. \tag{6}$$

ここに，$A_i(i=1,\cdots,n;n\geq 0)$及びBは一階述語論理のアトム(Atomic formula)である．$A_1 \wedge \cdots \wedge A_n$を前件，Bを後件，IDを節名と呼ぶ．

また，正規デフォルトは，$A_1 \wedge \cdots \wedge A_n : B / B$なる推論規則$(n\geq 0)$に限定し，以下の形式で表わす．

$$ID::A_1 \wedge \cdots \wedge A_n \rightarrow assume(B). \tag{7}$$

ここに，$A_1 \wedge \cdots \wedge A_n$を前提，Bを結論，IDをデフォルト名と呼ぶ．ただし，n=0の場合，以下の形式とする．

$$assume(B). \tag{8}$$

なお，形式(4),(5),(7),(8)においてBに含まれる全ての変数は$A_i(i=1,\cdots,n;n\geq 0)$中に出現していなければならないとする．

3・2　構成と動作の概略

本システムは，W_{-1}およびD_{-1}を空集合，$N\geq 0$とし，$(W_0 \cup \cdots \cup W_{N-1}, D_0 \cup \cdots \cup D_{N-1})$に対応するReteに類似したネットワーク（Rete-likeネットワークと呼ぶ）とその推論結果が既に存在し，(W_N, D_N)が新しく入力されたときに，$(W_0 \cup \cdots \cup W_N, D_0 \cup \cdots \cup D_N)$に対応するRete-likeネットワークを生成するインクリメンタル・コンパイラ(Incremental compiler)と，それが更新するネットワークに基づいて前向き推論を実行する推論エンジン(Inference engine)と推論結果の信念を管理するATMSとから構成されている(Fig.2)．

Rete-likeネットワークは，Reteネットワークと同様にルート・ノード，1入力ノード(One-input node)，2入力ノード(Two-input node)，終端ノード(Terminal node)およびそれらを連結するリンクから構成されているが，その目的が異なっており，ATMSのラベル計算を効率化[7]する．

インクリメンタル・コンパイラには，Rete-likeネットワークの構築・変更が行えるようにATMSを修正したモジュール（Rete-likeネットワーク構築用ATMSの意味で，以下RATMSと呼ぶ）を利用している．基本的な考えは，ATMSの環境束をRete-likeネットワークとみなせるということである．インクリメンタル・コンパイラに用いられるRATMSと実際の推論結果を管理するATMSとは構成上別モジュールである．

推論エンジンはトークンのキューを持ち，融合の結果新しくIN状態になった基礎アトムが連結される．推論エンジンは，キューから要素を取出しルート・ノードから流し，終端ノードに至るまでの途中の2入力ノードにおいて中間的な理由付けをATMSに送り，その結果を保存する．

ATMSは，推論エンジンによって基礎化代入されたホーン節(Ground Horn clause)および基底仮説(Ground hypothesis)の集合を入力し，基礎アトムの信念の状態を管理して出力する．

3・3　インクリメンタル・コンパイラ

3・3・1　RATMS

ATMSに次の修正を施すことによってRATMSが容易に実現できる．まず，環境ノードのスロットとしてSuperとSubを追加し，環境束を表現する手続きをラベル計算アルゴリズムに組み込んでいる．生成された環境束をRete-likeネットワークとして推論時に用いるため，基礎式を保

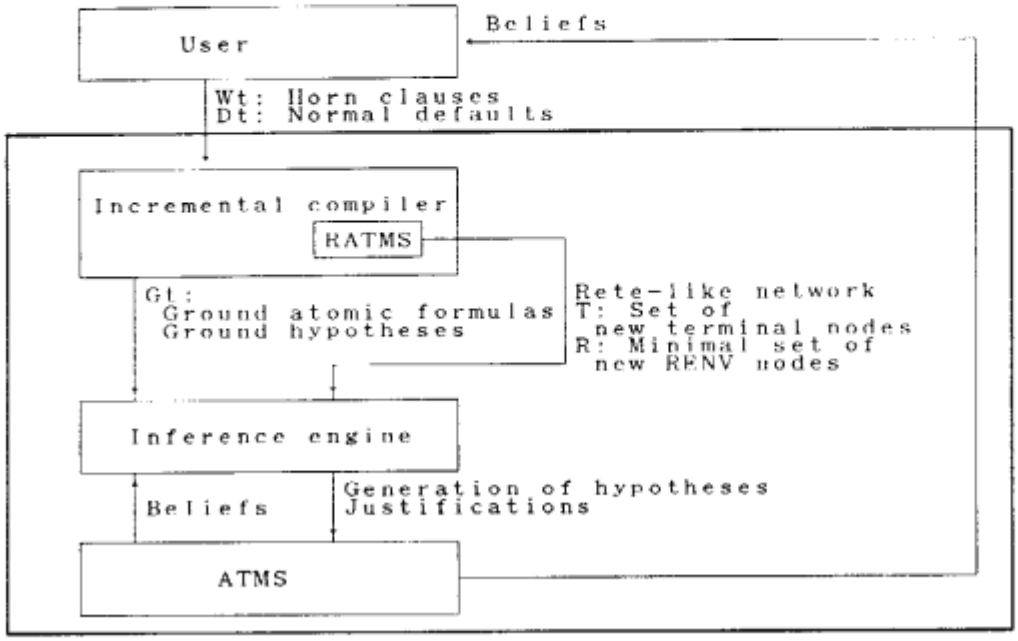


Fig. 2 Configuration of an incremental hypothesis-based forward-reasoning system.

存するワーキング・メモリとしてのWMスロットを環境ノードのスロットとして設けている．

すなわち，ATMSの環境ノードのデータ構造に以下のスロットを追加する．

Super：上位環境ノードへのポインタ（Super linkと呼ぶ）の集合（全ての上位環境ノードではない）．

Sub：下位環境ノードへのポインタ（Sub linkと呼ぶ）の集合（全ての下位環境ノードではない）．

WM：Reteノード・メモリ（推論実行時に使用され基礎式の集合が保存される）．

したがって，環境ノードは，以下のようなデータ構造となる．

<Assumptions, Nodes, Super, Sub, WM>.

また，Fig.1に示した手続きEnvironmentsの処理(*)を以下の処理とする．

let ε_i' be $\langle V_i', [], [], [\varepsilon_{i-1}', \varepsilon_i], [] \rangle$;

insert ε_i' in Super of ε_{i-1}' ;

insert ε_i' in Super of ε_i;

このように修正すると，ε_i' は２入力ノードとみなせる．

以下，RATMSでの環境ノードをRENVノード，RATMSでのATMSノードをRATMSノード（データ構造はATMSノードと同様）と呼ぶ．

３・３・２　ＲＡＴＭＳによるネットワークの更新

初期設定として，以下の理由付けがRATMSに送られる．

⇒root.

このとき生成されるRENVノードε_0をルート・ノードと同一視する．

インクリメンタル・コンパイラに（W_N, D_N）が入力されたとき，以下の処理を行う．

まず，初期設定として，新しく生成された終端ノードの集合を格納するためのＴと新しく生成されたRENVノードの集合を格納するためのＲを空とする．

W_Nの各々の要素およびD_Nの各々の要素について以下の処理を行い，ルート・ノードε_0，新しい終端ノードの集合Ｔ，新しいRENVノードの極小集合Ｒおよび次に述べる集合G_Nを推論エンジンに出力する．

W_N中の任意の節が形式(5)であればG_Nにその節を追加する．また，D_N中の任意のデフォルトが形式(8)であれば，G_Nにそのデフォルトを追加する．そうでなければ，すなわち形式(4)，(6)の節または形式(7)のデフォルトであったならば，これをκとして，その前件または前提の各アトムA_iをフリーズ(Freeze)したデータの仮説を生成するタスクをRATMSに送る．

フリーズとは，ある式に含まれる全ての変数を新しい定数記号に置き換えることを意味する．新しい定数記号として，例えば，$を用いる．さらに複数個の変数が含まれるときには$の後に自然数を付加して区別することとする．その操作を行う関数をＦとする．このようにして扱う理由は次のようである．RATMSノードをパターン・マッチングによって検索するRATMSは，変数を含むデータを登録すると次のような不都合が生じる（ATMSも同様である）．例えば，a(X)とa(1)とはマッチするが，同一のRATMSノードとしては扱えない．変数を含むデータをフリーズして扱えば，Ｆ（a(X)）＝a($)とa(1)とはマッチせず，異なったRATMSノードが生成される．

いま，$\Gamma_{F(Ai)}$が新しい仮説であれば新しいRENVノードεが生成されてＲに追加される．このとき，ε_0のSuperスロットにεへのポインタを追加し，このεを１入力ノードとみなす．RENVノードはビットベクタで一元管理されているので，生成される１入力ノードは複数の知識で共有可能ならば共有される．

ただし，ある一つの節の前件またはある一つのデフォルトの前提の各アトムA_i(i=1,…,n)をフリーズしたデータがその節の前件またはそのデフォルトの前提の他のアトムA_j($i \neq j$)をフリーズしたデータと等しいときには，$\Gamma_{F(Ai)}$と$\Gamma_{F(Aj)}$に対応する別々のRATMSノードを生

成することとする．これは，以下に述べる理由付けの結果，SuperリンクおよびSubリンクのループを避けるためである．

次に，κの前件あるいは前提の各アトムをフリーズしたデータの仮説を前件として，κをフリーズしたデータを後件とするような以下の理由付けをRATMSに送る．

$$\Gamma_{F(A1)}, \cdots, \Gamma_{F(An)} \Rightarrow F(\kappa).$$

RATMSでは，この理由付けに伴うラベル計算を行う．前節の修正により，ラベル計算の中間過程で生成されるRENVノードは２入力ノードとみなせる．RENVノードは一元管理されているので，生成される２入力ノードは複数の知識で共有可能ならば共有される．このとき新しく生成されたRENVノードはＲに追加される．また，この理由付けの後件を終端ノードとみなす．新しい終端ノードであるF(κ)は，Ｔに追加される．

W_NおよびD_Nの全ての要素の処理が終了した時点で，Ｒを極小化する．また，Ｔの各々の要素（終端ノード）に対応するRATMSノードのLabelの要素がＲの任意の要素の上位環境ノードであれば，その終端ノードをＴから削除する．

以上のように，RENVノードを１入力ノードまたは２入力ノードと，仮説ノード以外のRATMSノードを終端ノードとそれぞれ同一視する．

Fig.3は，W_0={r::a(X)∧b(Y)∧c(Z)→d(X,Y,Z)}，D_0=｛｝なる入力に対応するRATMSの出力を例示するものである．

３・４　インクリメンタル推論アルゴリズム

推論エンジンはトークンのキュー（Queue）を持ち，それに新しくIN状態になった基礎アトムが追加される．初期設定として，このQueueを空にする．

いま，（$W_0 \cup \cdots \cup W_{N-1}$，$D_0 \cup \cdots \cup D_{N-1}$）のもとで信じられている基礎アトムが全て求められATMSに保存されているとし，このシステムに（W_N，D_N）が，入力されたものとする．（W_N，D_N）は，先に述べたインクリメンタル・コンパイラにより，（$W_0 \cup \cdots \cup W_{N-1}$，$D_0 \cup \cdots \cup D_{N-1}$）に対応するRete－likeネットワークと共有をとる形でコンパイルされる．

推論エンジンは，コンパイラにより前述のルート・ノードε_0，新しい終端ノードの集合Ｔ，新しいRENVノードの極小集合ＲおよびWN中の形式(5)の節とDN中の形式(8)のデフォルトとからなる集合GNを入力し，Fig.4(a)に示す主動作アルゴリズムに基づいて推論を行い，（$W_0 \cup \cdots \cup W_N$，$D_0 \cup \cdots \cup D_N$）のもとで信じうる基礎アトムを全て求めATMSに保存させる．

［ステップ１］　まず，Fig.4(b)に示す手続きNewRENVを実行する（それらの処理が終了した後ステップ７へ）．Ｒの要素は，新しい１入力ノードに対応するRENVノードである（ステップ２へ）か，新しい２入力ノードに対応するRENVノードである（ステップ３へ）かのいずれかである．

［ステップ２］　新しい１入力ノードに対応するRENVノードである場合，そのパターンとマッチする現時点で信じられている基礎アトムが既に存在している可能性がある．この場合，そのRENVノードのSuperリンクで指示されるRENVノードに既に信じられている基礎アトムをトークンとして流す必要がある．そこで，現時点において信じられているデータτをトークンとして，以下の１入力ノードに関する処理を行う（ステップ４へ）．

［ステップ３］　新しい２入力ノードに対応するRENVノードであれば，Ｒの極小性により，そのSubリンクで指示されるRENVノードは必ず以前から存在していたことになる．その以前から存在していたRENVノードのWMの信じられている各々の要素をトークンとしてその新しい２入力ノードに対応するRENVノードに流す（ステップ５へ）．

［ステップ４］　１入力ノードに関する処理として，Fig.4(c)に示す手続きOneInputNodeを実行する．そのRENVノードのNodesリンクで指示されるRATMSノードのDatumをメルトする．メルト(Melt)はフリーズの逆関数で＄で始まる定数記号に新たな変数を代入する操作を行う．ただ

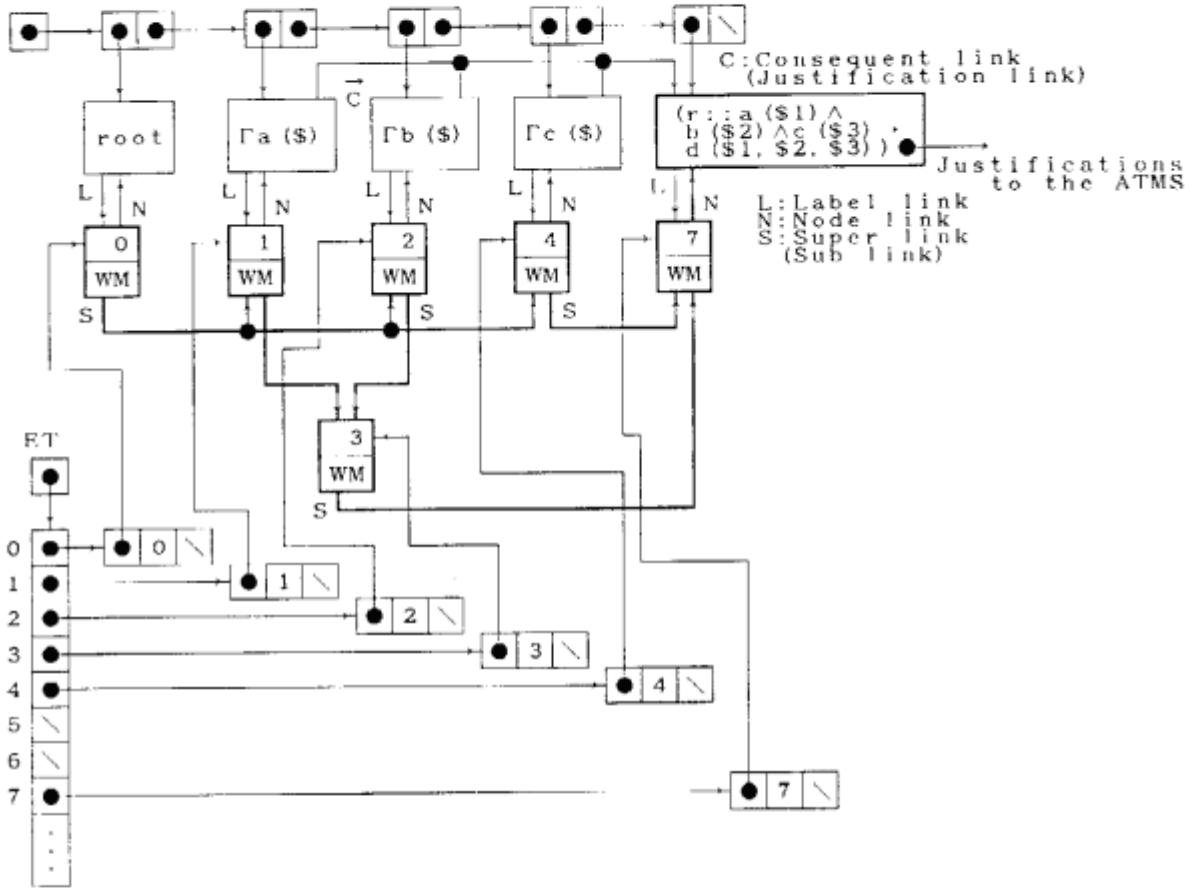


Fig. 3 Example of the output of RATMS.

し，同一の定数記号に対しては同一の変数を割り当てることとする．また，メルトの関数記号をMで表すものとする．メルトしたデータがΓ_Aの形式であり，アトムAがトークンτとマッチすれば，τをそのRENVノードが指示する終端ノードに流し（ステップ６へ）た後，次の処理を行う．もし，τが依然信じられていればWMにτを連結し，そのRENVノードが指示する２入力ノードに流す（ステップ５へ）．

［ステップ５］　２入力ノードに関する処理として，Fig.4(d)に示す手続きTwoInputNodeを実行する．そのRENVノードのSuperリンクで指示される各々のRENVノードε′にトークンτが流れてきたときに，そのSubリンクで指示される各々のRENVノードのWMの信じられている各々の要素ωについて，以下の中間的な理由付けをATMSに送る．

$$\tau,\ \omega \Rightarrow \tau \wedge \omega .$$

その結果の$\tau \wedge \omega$が信じられていれば，$\tau \wedge \omega$をトークンとして，そのRENVノードが指示する終端ノードに流し（ステップ６へ）た後，次の処理を行う．もし，$\tau \wedge \omega$が依然信じられていればWMにそれを連結し，そのRENVノードが指示する２入力ノードに流す（ステップ５へ）．

「ステップ６］　終端ノードに関する処理として，Fig.4(c)に示す手続きTerminalNodeを実行する．そのRENVノードのNodesリンクで指示されるRATMSノードのDatumをメルトする．メルトされたデータの形式を識別し，その前件または前提がトークンτとマッチすれば，対応する理由付けをATMSに送る．その理由付けの後件が新しいデータであれば，Queueの末尾に連結する．

［ステップ７］　また，前件や前提が全く同じ節やデフォルトが追加された場合は，新しいRENVノードは生成されないが，その追加知識により新しい基礎アトムがIN状態になる可能性がある．そこで，これに関する処理Fig.4(f)に示す手続きNewTerminalを実行する（それらの処理が終了した後ステップ８へ）．これは，Tの要素のLabelの要素であるRENVノードのWMの要素ωが信じられていれば，ωをトークンとして，そのRENVノードが指示する終端ノード（ステップ６へ）に流すことを示す．

［ステップ８］　Fig.4(a)に示す残りの手続きを実行する．すなわち，G_Nの要素に対応する理由付けをATMSに送り，その後件をQueueの末尾に連結する．次に，Queueの先頭から要素を取出しそれをトークンとして，ネットワークのルート・ノードから１入力ノードへ流す（ステップ４へ）．これをQueueが空になるまで行う．

４．評価

４・１　例題

段階的前向き仮説推論システムの評価を論理回路設計問題解決を例に行う．問題は，「２つの整数の最大公約数を計算する機能を持ちチップ面積（基本セル数がある値以下）と性能（遅延時間がある値以下）の制約を満たす回路を設計せよ[17]」とする．ここでは，基本セル数が400以下，遅延時間が60[ns]以下とする．

システムへの入力は，データパス設計，コンポーネント設計，テクノロジー・マッピング(CMOS)に関する知識および制約知識[17]を表現した節やデフォルトとする[18]．ここに，データパス設計知識を増やして段階的に問題解決を行うものとする．

以下に，実験のケースと入力知識ベースの状態を示す．

ケース１：W_0は全てのテクノロジー・マッピング知識，全ての制約知識を表現した節の集合とする．D_0は，全てのコンポーネント設計知識とある１つのデータパス設計知識を表現したデフォルトの集合とする．

ケース２：（W_1，D_1）をケース１に追加する．

ケース３：（W_2，D_2）をケース２に追加する．

ケース４：（W_3，D_3）をケース３に追加する．

```
procedure Reason (G_N, ε_0, T, R, Queue, ATMS) :
begin
NewRENV (R, Queue, ATMS) ;
NewTerminal (T, Queue, ATMS) ;
for all κ∈G_N do
 if κ=B where B is a ground atomic formula then
  begin
  give (⇒κ) to ATMS;
  concatenate κ to the end of Queue
  end
 else
  begin
  comment κ=(assume (B)) ;
  give (Γ_β⇒B) to ATMS;
  concatenate B to the end of Queue
  end;
while Queue is not empty do
 begin
 let τ be the first element on Queue;
 move τ from Queue to the root node ε_0;
 for all ε in Super of ε_0 do
  OneInputNode (ε, τ, Queue, ATMS)
 end
end.
```

(a) Main procedure to reason on the Rete-like network.

```
procedure NewRENV (R, Queue, ATMS) :
for all ε∈R do
 if ε is a one-input node then
  for all data τ in ATMS do
   OneInputNode (ε, τ, Queue, ATMS)
 else
  begin
  comment ε is a two-input node;
  let ε' be an element of Sub of ε;
  for all believed data ω in WM of ε' do
   TwoInputNode (ε, ω, Queue, ATMS)
  end.
```

(b) Procedure on new two-input nodes

```
procedure OneInputNode (ε, τ, Queue, ATMS) :
begin
 for all η in Nodes of ε do
  begin
   let δ be Datum of η;
   if M(δ) = (Γ_A) then
    if ∃θ (Aθ=τ) then
     begin
     TerminalNode (ε, τ, Queue, ATMS) ;
     if τ is believed then
      begin
      insert τ in WM of ε;
      TwoInputNode (ε, τ, Queue, ATMS)
      end
     end
  end
end.
```

(c) Procedure on one-input nodes

```
procedure TwoInputNode (ε, τ, Queue, ATMS) :
begin
 for all ε' in Super of ε do
  for all ε" in Sub of ε' do
   if ε≠ε" then
    for all ω in WM of ε" do
     if ω is believed then
        begin
        give (τ, ω=τ∧ω) to ATMS;
        if (τ∧ω) is believed then
         begin
         TerminalNode (ε', τ∧ω, Queue, ATMS) ;
         if (τ∧ω) is believed then
          begin
          insert (τ∧ω) in WM of ε;
          TwoInputNode (ε', τ∧ω, Queue, ATMS)
          end
         end
        end
end.
```

(d) Procedure on two-input nodes

```
procedure TerminalNode (ε, τ, Queue, ATMS) :
for all η in Nodes of ε do
 begin
 let δ be Datum of η;
 if M(δ) = (ID::A1∧…∧An→B) then
  if ∃θ ((A1∧…∧An)θ=τ) then
   begin
   give (A1θ, …, Anθ⇒Bθ) to ATMS;
   if Bθ is a new datum then
    concatenate Bθ to the end of Queue
   end
 else
  if M(δ) = (ID::A1∧…∧An→⊥) then
   if ∃θ ((A1∧…∧An)θ=τ) then
    give (A1θ, …, Anθ⇒⊥) to ATMS
  else
   begin
   comment M(δ) = (ID::A1∧…∧An→assume (B)) ;
   if ∃θ ((A1∧…∧An)θ=τ) then
    begin
    give (A1θ, …, Anθ, Γβθ⇒Bθ) to ATMS;
    if Bθ is a new datum then
     concatenate Bθ to the end of Queue
    end
   end
 end.
```

(e) Procedure on terminal nodes

```
procedure NewTerminal (T, Queue, ATMS) :
for all η∈T do
 for all ε in Label of η do
  for all ω in WM of ε do
   if ω is believed then
    TerminalNode (ε, ω, Queue, ATMS).
```

(f) Procedure on new terminal nodes

Fig. 4 Incremental reasoning algorithm.

ケース５：（W_4，D_4）をケース４に追加する．
W_1からW_4は全て空集合である．D_0中の１つのデータパス設計知識を表現したデフォルトおよびD_1からD_4をFig.5に示す．

４・２　評価方法

段階的前向き仮説推論システムのコンパイル時間と推論時間について評価する．

比較の基準とするシステムは，前向き推論エンジンとATMSとを単純に結合した仮説推論システム(Simple Combination System;SCSと呼ぶ)[7][18]である．

SCSの段階的推論の方式について概要を示す．まず，初期入力知識ベース（W_0，D_0）があるとする．（W_0，D_0）各々の要素をESP[13]のローカル節と１対１対応にコンパイルする．コンパイル結果をP_0とし，P_0はATMSと交信しながら前向き推論を実行する．その結果，IN状態となる基礎アトムが全て（この集合を１とする）ATMSに保存されている．その後から追加される入力知識ベースを（W_1，D_1）とすると，SCSのコンパイラも追加入力知識ベース（W_1，D_1）だけをインクリメンタルにコンパイルして，（W_1，D_1）に対応するESPオブジェクトP_1を出力する．このとき，P_0とP_1とを用いてＩから推論を行う．以下，同様とする．

４・３　評価結果

Fig.6に，本インクリメンタル・コンパイラを用いて生成したRete－likeネットワーク例を示す．図(a)は，デフォルトdatapath_1だけに関するものである．図(b)は，ケース５に対応する知識ベースに含まれるデータパス設計知識を表現した５つのデフォルトのみに関するものである．図中，ボックスの中にビットベクタを十進表現した数を記しRENVノードを示す．ただし，Nodesスロットの要素が仮説ノードや終端ノードを示す場合，そのパターン（対応するアトム）や終端ノード（デフォルト名の頭部に記号#を付けて対応するデフォルトの終端ノードを表示）をボックスの中に記しRENVノードを示す．また，同じデータのRATMSノードが複数存在する（例えば，register(A,B),A,Bは変数）が，それらのLabelの要素のRENVノードについては１つだけ示している．

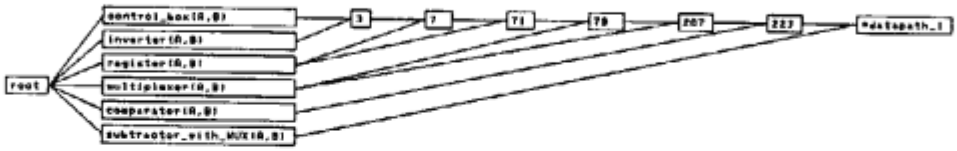
datapath_2は，datapath_1とε_3からε_{223}までを共有した構造になっている．datapath_3も同様である．datapath_4は，それらの知識とε_3からε_{207}までを共有した構造になっている．datapath_5も同様である．

また，インクリメンタルな推論実行について，ケース２を例に説明する．新しく生成されたRENVノードの集合の極小集合は，「subtracter_with_CMPL($1,$2)に対応するRATMSノードのLabelの要素のRENVノードのみからなる集合である．そのパターンとマッチするIN状態の基礎アトムをそのRENVノードのWMスロットに追加して，その基礎アトムをトークンとしてそのSuperリンクで指示されるRENVノードに送る．そこでは，以前の推論処理で既にラベル計算の終わっているε_{223}のWMの要素と連言がとられ中間的な理由付けがATMSに送られる．その中間的なデータはデフォルトdatapath_2の終端ノードに送られる．ケース３からケース５においても，RENVノードの共有が存在し，入力知識ベースの変更前に行われたラベル計算の中間結果を有効に活用して，変更後の推論を効率的に行う．

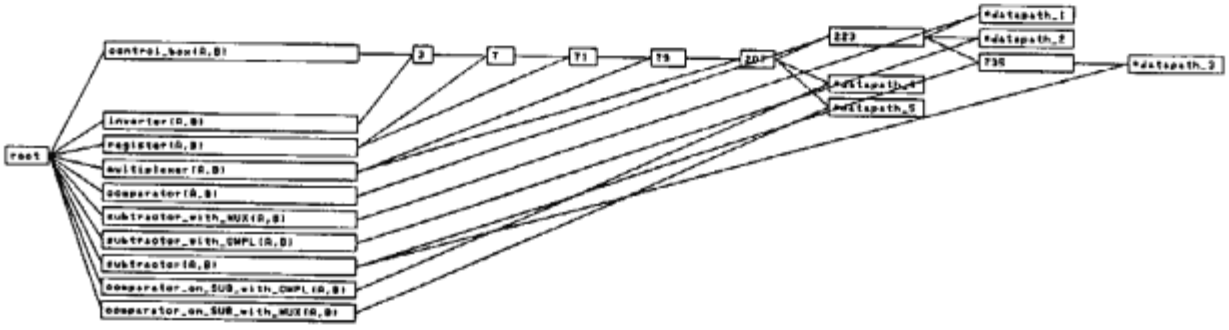
Table 1に，PSI－II[12]マシン上のインクリメンタルなコンパイル時間T_cと推論時間T_rとに関する本システムのデータとSCSのデータとを示す．実験結果によると，ケース１で，本システムの方が約３割コンパイル時間が長い．しかし，ケース２からケース３では，ほとんどコンパイル時間の差はない．一方，それらのコンパイル結果を用いた推論時間については，ケース１で，本システムの方が５倍高速である．また，ケース２からケース５の結果で，入力知識ベースが変更されるたびに本システムの推論実行効率が高くなっていく（１２倍から５９倍程度）ことがわかる．これは，SCSにおいては入力知識ベースの変更を繰り返すと変更前の推論

```
D0= { ..............
     datapath_1::
      control_box (X1, N1) ∧inverter (X2, N2) ∧
      register (X3, N3) ∧register (X4, N4)
      multiplexer (X5, N5) ∧multiplexer (X6, N6) ∧
      comparator (X7, N7) ∧subtracter_with_MUX (X8, N8) →
      assume (calculator_of_GCD ([X1, X2, X3, X4, X5, X6, X7, X8],
                                 N1+N2+N3+N4+N5+N6+N7+N8) )} .

D1= {datapath_2::
      control_box (X1, N1) ∧inverter (X2, N2) ∧
      register (X3, N3) ∧register (X4, N4) ∧
      multiplexer (X5, N5) ∧multiplexer (X6, N6) ∧
      comparator (X7, N7) ∧subtracter_with_CMPL (X8, N8) →
      assume (calculator_of_GCD ([X1, X2, X3, X4, X5, X6, X7, X8],
                                 N1+N2+N3+N4+N5+N6+N7+N8) )} .

D2= {datapath_3::
      control_box (X1, N1) ∧inverter (X2, N2) ∧
      register (X3, N3) ∧register (X4, N4) ∧
      multiplexer (X5, N5) ∧multiplexer (X6, N6) ∧
      comparator (X7, N7) ∧subtracter (X8, N8) ∧
      subtracter (X9, N9) →
      assume (calculator_of_GCD ([X1, X2, X3, X4, X5, X6, X7, X8, X9],
                                 N1+N2+N3+N4+N5+N6+N7+N8+N9) )} .

D3= {datapath_4::
      control_box (X1, N1) ∧inverter (X2, N2) ∧
      register (X3, N3) ∧register (X4, N4) ∧
      multiplexer (X5, N5) ∧multiplexer (X6, N6) ∧
      comparator_on_SUB_with_CMPL (X7, N7) →
      assume (calculator_of_GCD ([X1, X2, X3, X4, X5, X6, X7],
                                 N1+N2+N3+N4+N5+N6+N7) )} .

D4= {datapath_5::
      control_box (X1, N1) ∧inverter (X2, N2) ∧
      register (X3, N3) ∧register (X4, N4) ∧
      multiplexer (X5, N5) ∧multiplexer (X6, N6) ∧
      comparator_on_SUB_with_MUX (X7, N7) →
      assume (calculator_of_GCD ([X1, X2, X3, X4, X5, X6, X7],
                                 N1+N2+N3+N4+N5+N6+N7) )} .
```

Fig. 5 Defaults representing datapath design knowledge.



(a) In Case 1.



(b) In Case 5.

·One-input nodes are denoted by boxes with melted patterns.

·Two-input nodes that have no terminal node are denoted by boxes with numbers.

·Two-input nodes that have a terminal node corresponding to a clause or a default named ID are denoted by boxes with #ID.

Fig. 6 Rete-like network corresponding to the defaults representing datapath design knowledge.

Table 1 Evaluation of the incremental hypothesis-based forward-reasoning system.

·Tc: Compiling time on the incremental hypothesis-based forward-reasoning system.

·Tr: Reasoning time on the incremental hypothesis based forward-reasoning system.

·Tc': Compiling time on SCS.

·Tr': Reasoning time on SCS.

Case	Number of total caluses	Number of total defaults	Number of solutions	Tc [s]	Tr [s]	Tc' [s]	Tr' [s]	Ratio Tr'/Tr
1	44	5	0	16.4	1.4	12.8	6.5	5
2	44	6	0	1.1	0.3	1.1	3.7	12
3	44	7	0	1.1	0.4	1.1	7.1	18
4	44	8	0	1.0	0.8	1.0	24.2	30
5	44	9	2	1.0	0.5	1.0	29.3	59

と重複した計算が避けられないのに対して，本システムではそれをうまく避けることができることによる．さらに，それぞれのケースにおいて，T_cとT_rとの和は，常に本システムの方がSCSに比べ短い．以上，本システムは，段階的仮説推論を効率的に行うことが確認された．

5． 他の研究との関連

5・1 Reteアルゴリズムとの相違点

OPS5[14]等の連言的パターン・マッチングの効率化のために開発されたReteアルゴリズムは，ネットワーク内にパターン・マッチングの中間結果を保存し，ワーキング・メモリの変更分のみをトークンとしてネットワークに流すものである．これにより，複数のルールによって1入力ノードや2入力ノードが共有される場合，重複したパターン・マッチング処理を避けることができる．いわば，変数間の束縛値に関する制約評価を部分的に行いその中間結果を保存するのが，Reteアルゴリズムである．

Rete-likeネットワークを用いる目的は上記と異なり，ATMSを用いた前向き仮説推論システムにおけるATMSの全ラベル計算コストの減少にある．

RATMSを用いたコンパラが出力するネットワークは，パターン・マッチングの効率化という点に関しては，不十分な点がある．それは，パターン・マッチング処理を1入力ノードと終端ノードのみで行っており，2入力ノードではそれを行わないようになっているからである．しかしながら，この点は，ラベル計算の効率化という点からすれば，利点となることも多い．

例えば，2つ以上のアトムの変数間の束縛に関する制約がある知識と，その制約のないアトムの連言がある知識が存在する場合を考える．その制約を終端ノードで評価することにすれば，制約のあるアトムの連言の基礎式のラベル計算は，制約のないアトムの連言の基礎式の集合のある要素のラベル計算結果として利用可能である．すなわち，ラベル計算が共有できる．これに対して，2入力ノードをアトムの変数間の束縛に関する制約を含めて考えると，制約のあるアトムの連言に対応する2入力ノードと制約のないアトムの連言に対応する2入力ノードが別々になってしまい，そのラベル計算は共有されなくなってしまう．

［具体例］

$$W_0=\{r1::a(X)\wedge b(Y)\rightarrow d(X,Y),\ r2::a(X)\wedge b(X)\rightarrow c(X)\},$$
$$D_0=\{assume(a(1)),assume(b(1)),assume(b(2))\}$$

なる知識ベースがあったとする．このとき，本システムにおいては，$a(X)\wedge b(Y)$に相当する2入力ノードが1個だけ存在する．この2入力ノードにおいて，以下の理由付けが送られる．

$$J1:a(1),b(1)\Rightarrow a(1)\wedge b(1).$$
$$J2:a(1),b(2)\Rightarrow a(1)\wedge b(2).$$

このように，2入力ノードを1つにしておくことにより，J1による中間的なラベル計算が節r1,r2で共有できる．

5・2 ネットワーク構造に関する考察

以下に，本論文で示したインクリメンタル・コンパイラによって生成されるネットワークの構造について考察する．Fig.1のアルゴリズムを用いると左優先の線形Reteネットワーク[15]になる．ラベル計算の制御を変えることにより，バイナリReteネットワーク[15]や一般化されたReteネットワーク構造[10]とほぼ同様な構造が得られる．

また，変数をフリーズして先のような理由付けを受けるRATMSは，Justificationsリンクおよび Consequentsリンクで構成されるネットワークをも同時に作成する．このネットワークは，複数の1入力ノードからのリンクが終端ノードでマージされ，2入力ノードがないという性質をもつ．そこで，このネットワークを利用する解釈器を構成すれば，それはTREAT[16]アルゴ

リズムと似た推論エンジンとなる．

6．むすび

以上，前向き推論エンジンとATMSとを結合した仮説推論システムにおけるインクリメンタル・コンパイラの実現方法とその出力を扱う推論アルゴリズムを示した．これらによって構成できる段階的前向き仮説推論システムは，変数を含む知識を段階的に追加できるようにATMSを拡張したものと考えられる．この仮説推論システムは，今までに与えられた知識と矛盾するかもしれない新しい変数を含む知識を後から追加する状況が頻繁に起こり得るような知識システムの効率的なツールとして用いることができる．また，論理回路設計問題を例に実験を行った結果，特に入力知識ベースが変更された後の推論実行効率が極めて良いことが確認された．

なお，RATMSを用いて生成されるRete-likeネットワークにおいて，ノードが入力知識の増加に伴い単調に増加するので，その削除の技術が必要と考えられる．この点に対して，RATMSにNogoodを送ることによりノードを削除することができる[6][11]．しかしながら，このようにして削除された知識はNogoodデータベースに残るために再度追加することができない．したがって，入力知識が頻繁に削除・追加される状況に対応するためには，柔軟にネットワークを変形操作するための方法が課題として残される．また，RATMSを用いて生成されるネットワークを最適化[19]する方法についても今後検討したい．この点に関しては，全ての仮説の組合せに対応する環境ノードの集合とその要素間の上位および下位関係を考えた完全束上で，全ての終端ノードのラベルの要素の下位環境ノードを取出し，一番共有度の高い環境ノードを２入力ノードとみなすと，ある意味で最適なネットワーク構造を生成できると考えられる．

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，第五研究室 長谷川隆三室長に深く感謝致します．また，御指導頂いたNTTデータ　生駒憲治氏，NTT藤井裕一氏，本論文をまとめるにあたり多くの貴重なコメントを頂いたICOT古川康一次長，論理回路設計問題領域から貴重なコメントを頂いた富士通研究所 丸山文宏氏，論理回路合成問題への適用実験に御協力頂いたJIPDEC大崎宏氏ならびに中島誠氏に深く感謝致します．さらに，討論して頂いたICOT研究員の方々に感謝します．

E

◇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)Reiter,R.: A Logic for Default Reasoning,Artif.Intell.,Vol.13,pp.81－132(1980).
(6)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).
(7)太田好彦，井上克巳：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，人工知能学会誌，Vol.6,No.2,掲載予定(1991).
(8)Forgy,C.,L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem,Artif.Intell.,Vol.19,pp.17－37(1982).
(9)荒屋真二，百原武敏，田町常夫：知識ベースの逐次構造化－Reteネットワークの逐次構築法－，信学論D,Vol.J71－D,No.6,pp.1100－1108 (1988).
(10)Lee,H.,S. and Marshall,I.,S.: Dynamic Augmentation of Generalized Rete Networks for Demand－Driven Matching and Rule Updating,Proc.of the Sixth Conference on Artificial Intelligence Applications,pp.123－129(1990).
(11)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).
(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)Brownston,L.,Farrell,R. and Elaine,K.: Programming Expert Systems in OPS5, Addison－Wesley(1985).
(15)Gupta,A.: Parallelism in Production Systems,PhD thesis,Department of Computer Science,Carnegie Mellon University(1985).
(16)Miranker,D.,P.: Performance Estimates for the DADO Machine: A Comparison of TREAT and RETE,Proc.of the International Conference on Fifth Generation Computer Systems,pp.449－457(1984).
(17)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).
(18)Ohta,Y. and Inoue,K.: A Forward－Chaining Multiple－Context Reasoner and Its Application to Logic Design,Proc.of the 2nd International IEEE Conference on Tools for Artificial Intelligence,pp.386－392(1990).
(19)石田 亨： プロダクションシステムにおける条件記述の最適化，情報処理学会論文誌，Vol.29,No.12,pp.1158－1169(1988).