ICOT Technical Report: TR-555

TR-555

ATMSを用い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

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，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]．この仮説推論の高速化について，文献（3）では，ATMS[4]と前向き推論システムとを結合した仮説推論システムについて述べられている．これは，正規デフォルト[5]とホーン節によって定義される入力知識ベースから前向き推論によって得られる全ての基礎アトムとそれらのラベルを得るという機能に関し，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方式により効率的な推論を実現している．この方式による高速化の効果は，2入力ノードが複数の知識に共通である場合または2入力ノードの後続2入力ノードに1入力ノードから複数のトークンあるいは複数の環境を有するトークンが流れ込む場合に有効であると結論付けられている．

ところで，Reteアルゴリズム[6]は，推論実行に先立ち，複数のルール間で共通のパターンを見出し，無駄なパターン照合を行わないように，与えられたルール・セットをネットワーク構造にコンパイルしておく方法である．よって，ルール・セットの一括コンパイルが前提条件となっている．しかしながら，実用的な知識システムは開発時等において頻繁にルール・セットに部分的修正が加えられるため，一括コンパイルの前提には問題がある．特に，大規模な知識システムにおいては，Reteネットワークへのコンパイル時間は膨大になってしまう．荒屋ら[7]は，プロダクション・システムを対象に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．

本論文では，仮説推論システムにおいて，知識ベースからReteネットワークへのATMSを利用したインクリメンタル・コンパイラの実現方法について述べる．従来のReteネットワークのインクリメンタル・コンパイラと比較した本コンパイラの特徴は，ATMSに修正を施して容易に実現できること，削除された知識に関する情報を有効に保存し入力知識ベースの管理を行うこと等が挙げられる．

更に，仮説推論システムAPRICOT／0[3][8]のインクリメンタル・コンパイラを本実現方法により，

ＰＳＩ－ＩＩ[9]マシン，言語ＥＳＰ[10]上にインプリメントし，本方法の有効性を確認した．本論文は，ＡＰＲＩＣＯＴ／０の特徴のうち，文献（３）で示されている仮説推論の高速化以外の特徴，すなわち入力知識ベース管理機能を備えたインクリメンタル・コンパイラの実現方法について述べている．

２章においてＡＴＭＳの概略，３章においてＡＰＲＩＣＯＴ／０の概略，４章において本インクリメンタル・コンパイラの概要について述べ，５章においてそのコンパイラのネットワーク・ビルダの実現方法，６章でそのコンパイラの制御部の動作，７章で評価について詳細に述べる．

２．ＡＴＭＳの概略

ＡＴＭＳが無矛盾性を管理するデータは，命題論理のアトムのレベルである．データに対応してＡＴＭＳ内部では，＜データ，ラベル，理由付け＞なるデータ構造のノード（これを単に＜データ＞と略記することもある）が生成される．

命題論理のアトムA_i（$i = 1, \cdots, n ; n \geq 0$）の連言を前件とし，命題論理のアトムＢを後件としたホーン節（１）に基づきノード＜A_i＞（$i = 1, \cdots, n ; n \geq 0$）からノード＜Ｂ＞が導かれたことを理由付け（２）という形式でＡＴＭＳに与える．

$A_1, \cdots, A_n -> B.$　　（１）

$<A_1>, \cdots, <A_n> \Rightarrow <B>.$　　（２）

ここに，－＞は含意，＜A_1＞，…，＜A_n＞をこの理由付けの前件ノード，＜Ｂ＞をこの理由付けの後件ノードと呼ぶ．以下，ホーン節（１）は理由付け（２）に対応するという．

正規デフォルト：Ｂ／Ｂに相当する理由付けを以下のように表す．

$<\Gamma B> \Rightarrow <B>.$　　（３）

ここに，ΓＢを仮定，＜ΓＢ＞を仮定ノード，＜Ｂ＞を

ａｓｓｕｍｅｄノードと呼ぶ．

また，仮定の集合を環境と呼びＥで表す．環境は，ラベル計算の高速化のためにビットベクタ表現を採用し，以下のデータ構造である．また環境は，ビットベクタで一元的に管理されている．

ａｓｓｕｍｐｔｉｏｎｓ：ビットベクタ．
ｎｏｄｅｓ：ラベルの逆ポインタ．

３．仮説推論システムＡＰＲＩＣＯＴ／０の概略

仮説推論システムＡＰＲＩＣＯＴ／０は，入力知識ベースのコンパイラ[8][11]，推論エンジン及びＡＴＭＳから構成されている．入力知識ベースは，一階述語論理ベースであり，推論エンジンで具体化した理由付けを生成しＡＴＭＳに送る．ＡＴＭＳはこれにより基礎アトムに関する無矛盾性管理を行う．

入力知識ベースは，正規デフォルトとホーン節によって定義される．ホーン節は，正リテラルの数を高々１個に制限した節である．ここに，α_i（ｉ＝１，…，ｎ；ｎ≧０）及びβは一階述語論理のアトム（Ａｔｏｍｉｃ ｆｏｒｍｕｌａ）であり，ＩＤは節名であるとする．正リテラルの数が１の場合形式（４），負リテラルの数が０の場合形式（５），正リテラルの数が０の場合形式（６）で記述する．

ＩＤ：：α_1，…，α_n－＞β．　（４）

β．　（５）

ＩＤ：：α_1，…，α_n－＞［］．　（６）

また，正規デフォルトは，αをアトムα_i（ｉ＝１，…，ｎ；ｎ≧０）の連言として，$\alpha : \beta / \beta$なる推論規則に限定し，形式（７）で表わす．ここに，αを前提，βを結論，ＩＤをデフォルト名と呼ぶ．

ＩＤ：：α_1，…，α_n－＞ａｓｓｕｍｅ（β）．　（７）

：β / βは形式（８）で記述する．

assume（β）．　　（8）

4．本インクリメンタル・コンパイラの概要

4・1　Rete-likeネットワーク

ビットベクタ表現された環境の組み合わせ計算（ユニオンをとる操作）は，ビット演算orを用いて効率的に計算できる．通常，このとき，2つの環境ごとにビット演算orが実行される．従って，この点に着目し，Reteアルゴリズムの要旨である2入力ノードの実現をATMSの環境をReteノードとみなして容易に行えると考えられる．本論文で述べるインクリメンタル・コンパイラのネットワーク・ビルダは，ATMSに修正を施して極めて容易に実現される．

Reteネットワークは，Reteノードとそれらを結ぶリンクから構成される．Reteノードは，ルート・ノード，1入力ノード，2入力ノード，終端ノードに分類される．

Reteアルゴリズムは，ネットワークを用いて，パターン・マッチング処理の中間結果をReteノードのメモリ・ポインタを用いて保存しておき重複した処理を行わないようにする方法である．ここで，Reteネットワークの1入力ノードにはメモリ・ポインタがなく，2入力ノードにはメモリ・ポインタが2つある．リンクは全て一方向リンクである．本方法では，1入力ノードと2入力ノードをATMSの環境で表す．そこで，1入力ノードと2入力ノードのデータ構造を同様とするために，1入力ノードおよび2入力ノードにはメモリ・ポインタ（後述の分散ワークング・メモリに対応）を1つとし，リンクを双方向とする．これによって構成されたネットワークをRete-likeネットワークと呼ことにする．

Rete-likeネットワークには，トークンとも呼ばれる基礎式に対応するATMSノード（以下，単にノードと呼ぶこともある．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．

4・2　構成

コンパイラの構成をFig．1に示す．このコンパイ

ラは，入力知識ベースに対応するＲｅｔｅ－ｌｉｋｅネットワークを構築し出力すると共に知識ベースの変更に対する部分的修正を行うインクリメンタル・コンパイラである．尚，初期知識ベースのコンパイルは，知識を一つずつ追加していく処理とみなす．また，知識ベースの変更は，知識の追加と削除により行う．

インクリメンタル・コンパイラは，図示するように，パーサ，制御部，オブジェクト生成器及びネットワーク・ビルダにより構成されている．

ユーザが直接編集した知識ベースを追加するというタスクが与えられると，パーサは知識ベースに含まれる節やデフォルトを一つずつ読込むと共に構文解析し，知識の追加というタスクと共に解析結果を制御部に渡す．制御部は，形式（５）の節及び形式（８）のデフォルトを直接推論エンジンに送る．推論エンジンでは，対応するＡＴＭＳノードをトークン・キューに追加する．形式（４）及び（６）の節及び形式（７）のデフォルトは，オブジェクト生成器に送る．オブジェクト生成器では，Ｒｅｔｅノードのうちの終端ノードを生成し制御部へ返す．以下，＃ＩＤは節名ＩＤの節またはデフォルト名ＩＤのデフォルトに相当する終端ノードを示し，実行時にＡＴＭＳに理由付けを送る処理を行うプログラム単位（ＥＳＰのオブジェクト）とする．

更に，制御部は，形式（４）及び（６）の節及び形式（７）のデフォルトに対し，ネットワーク・ビルダから信念の状態とオブジェクト生成器により生成された終端ノードを用いて，６章で述べる理由付けをネットワーク・ビルダに送る．ネットワーク・ビルダは，ＡＴＭＳをコンパイラ用に修正を加えたモジュールである（詳細は，５章参照）．ネットワーク・ビルダは環境をＲｅｔｅノードとして取扱い，Ｒｅｔｅ－ｌｉｋｅネットワークを構築し，推論エンジンに送る．また，現在のネットワークの状態をユーザに表示することができる．

また，知識を削除するというタスクが与えられると，ユーザによって指定された削除する知識の前件または前提に含まれているアトムまたはそれらの組み合わせがパーサを通して制御部に送られる．制御部では，このアトムまたはそれらの組み合わせとネットワーク・ビルダから送られてくる信念の状態から理由付けを生成し，ネットワーク・ビルダへ送る．ネットワーク・ビルダでは，

削除のために不要になったＲｅｔｅノードやリンクを削除し，修正されたＲｅｔｅ－ｌｉｋｅネットワークを推論エンジンに送る．

５．ネットワーク・ビルダの実現方法

ＡＴＭＳをネットワーク・ビルダへ適用するため，以下の修正（環境のスロット及び操作の追加）を施す．ネットワーク・ビルダは入力知識ベースの管理を行い，Ｒｅｔｅ－ｌｉｋｅネットワークを構成するものである．

まず，ＡＴＭＳの環境のデータ構造に以下のスロットを追加し，これをＲｅｔｅノード（終端ノードを除く）として扱う．

ｓｕｐｅｒ：無矛盾な上位環境へのポインタの集合．
ｓｕｂ：下位環境へのポインタの集合．
ｗｍ：分散ワーキング・メモリ．推論実行時に使用する．基礎アトムに対応するＡＴＭＳノードへのポインタの集合．

Ｒｅｔｅノード間のリンクを実現するために，環境の組み合わせを生成するメソッドｏｒに以下の操作を追加し，ｓｕｐｅｒ，ｓｕｂリンクで表現されるようにする．すなわち，：ｏｒ（Ｅ１，Ｅ２，Ｅ３）のとき（このメソッドは，Ｅ１∪Ｅ２＝Ｅ３を意味する），以下に示す操作を追加する．

：ａｄｄ（Ｅ１！ｓｕｐｅｒ，Ｅ３），
：ａｄｄ（Ｅ２！ｓｕｐｅｒ，Ｅ３），
：ａｄｄ（Ｅ３！ｓｕｂ，Ｅ１），
：ａｄｄ（Ｅ３！ｓｕｂ，Ｅ２）．

ここに，：ａｄｄ（Ｏｂｊ１！Ｓｌｏｔ，Ｏｂｊ２）は，オブジェクトＯｂｊ１のスロットＳｌｏｔにオブジェクトＯｂｊ２を追加するという意味である．

また，Ｎｏｇｏｏｄ　ｄａｔａｂａｓｅに新たに矛盾環境Ｅが加えられたときに，スロットｓｕｂに含まれる全ての環境のスロットｓｕｐｅｒから環境Ｅを削除するという操作も追加する（リンクの切断のため）．

６．制御部の動作

６・１　初期設定

初期設定として，制御部は，理由付け（９）をネットワーク・ビルダに送り，内部においてノード（１０）を生成しておく．環境は，前記データ構造を持つが，以下その環境が保持するビットベクタを十進表現して記述する（この十進表現をＥの添字として，環境オブジェクトを示す）．ノード（１０）において０は仮定を１つも含まない環境をビットベクタ表現し，そのビットベクタを十進表現したものであり，環境０に対応するオブジェクトE_0がネットワークのルート・ノードとなる．

⇒＜ｐｒｅｍｉｓｅ＞．　（９）

＜ｐｒｅｍｉｓｅ，｛０｝，｛（）｝＞．　（１０）

６・２　知識の追加

一般に，ＡＴＭＳのデータは，命題のレベルでなければならない．そこで，ネットワーク・ビルダのために，ｆ（α_i）をアトムα_iに含まれる全ての変数を新しい定数記号（例えば$，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アトムを返す（フリーズする）関数（この逆関数をメルトと呼び，推論実行時に使用する）を考える．

制御部に知識の追加タスクが送られると，制御部は追加節の前件または追加デフォルトの前提の各アトムをフリーズしたデータのａｓｓｕｍｅｄノードを生成するタスクをネットワーク・ビルダに送る．すなわち，理由付け（１１）をネットワーク・ビルダに送る．

＜Γｆ（α_i）＞⇒＜ｆ（α_i）＞．　（１１）

＜Γｆ（α_i）＞が新しい仮定ノードであれば新しい環境Ｅが生成されて＜Γｆ（α_i）＞のラベルの要素となり，この環境Ｅが１入力ノードとして取り扱われる．環境は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される．新しい環境Ｅが生成されたとき，以下の操作を行う．

：ａｄｄ（Ｅ$_0$！ｓｕｐｅｒ，Ｅ），
：ａｄｄ（Ｅ！ｓｕｂ，Ｅ$_0$）．

次に，制御部は，追加節の前件または追加デフォルトの前提の各アトムをフリーズしたデータのａｓｓｕｍｅｄノードを前件ノードとして，後件を終端ノードとする理由付け（１２）をネットワーク・ビルダに送る．このとき，中間過程で生成される環境を２入力ノードとして取り扱う．環境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される．また，前述の環境の組み合わせを生成するメソッドｏｒに追加した処理により，Ｒｅｔｅノード間のリンクも張られる．

＜ｆ（α_1）＞，…，＜ｆ（α_n）＞⇒＜＃ＩＤ＞．
（１２）

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α_i（ｉ＝１，…，ｎ）をフリーズしたデータのａｓｓｕｍｅｄノード＜ｆ（α_i）＞の集合で，ｆ（α_i）＝ｆ（α_j）；ｉ≠ｊが成り立つ場合は５章で述べた追加処理によれば，リンクのループができてしまう．そこでこのような場合，ｆ（α_j）をｆ（α_j'）のように別のノードとして扱う．

例えば，節（１３）があり，前記原理に従い仮定の生成（１４）を行い，理由付け（１５）を送る．

ａｂｏｖｅ：：
ａｂｏｖｅ（Ｘ，Ｙ），ａｂｏｖｅ（Ｙ，Ｚ）
－＞ａｂｏｖｅ（Ｘ，Ｚ）．　（１３）

＜Γａｂｏｖｅ（＄，＄）＞
⇒＜ａｂｏｖｅ（＄，＄）＞．　（１４）

＜ａｂｏｖｅ（＄，＄）＞，＜ａｂｏｖｅ（＄，＄）＞
⇒＜＃ａｂｏｖｅ＞．
（１５）

ここで，Γａｂｏｖｅ（＄，＄）のラベルの要素をＥａとすると，後件のラベル計算のために（１６）が行われ

Ｅａのリンクのループができてしまう．そこで，このような場合別の仮定（１７）を考え，理由付け（１８）をネットワーク・ビルダに送る．

：ｏｒ（Ｅａ，Ｅａ，Ｅａ）．　　（１６）

＜Γａｂｏｖｅ′（＄，＄）＞
　　⇒＜ａｂｏｖｅ′（＄，＄）＞．　　（１７）

＜ａｂｏｖｅ（＄，＄）＞，
＜ａｂｏｖｅ′（＄，＄）＞
　　⇒＜＃ａｂｏｖｅ＞．　　（１８）

６・３　知識の削除

知識のインクリメンタルな削除は，削除する知識の前件または前提に含まれているアトムまたはそれらの組み合わせ［α_1，…，α_n］をユーザが指定する．制御部は，対応するａｓｓｕｍｃｄノードを前件ノードとし後件を偽とする理由付け（１９）をネットワーク・ビルダに送る．

＜ｆ（α_1）＞，…，＜ｆ（α_n）＞⇒＜⊥＞．　　（１９）

ネットワーク・ビルダは，今までに与えられた理由付け（１１）（１２）及び（１９）に対応するホーン節の集合Ｊ′（推論エンジンが生成する理由付けに対応するホーン節の集合Ｊとは独立の系）に対する矛盾環境をＲｅｔｅ－ｌｉｋｅネットワークから全て削除する（５章で述べた処理により，不要なリンクが切断される）．また，ネットワーク・ビルダのＮｏｇｏｏｄＤａｔａｂａｓｅにより，矛盾環境が極小で保存される．この後で，ユーザから知識の追加タスクが与えられると，制御部はその追加知識に対応する理由付け（（１１）（１２）に対応）をネットワーク・ビルダに送る．しかし，矛盾環境の基でのみ理由付け（１２）に対応するホーン節の後件＃ＩＤが成立するような知識ＩＤについては，ネットワーク・ビルダはネットワークに反映しない．

７．評価

７・１　Ｒｅｔｅ－ｌｉｋｅネットワークの構築例

本インクリメンタル・コンパイラ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人と月曜日または火曜日にミーティングを行いたい．部門１のａとｂ，部門２のｃとｄはこのミーティングの出席候補である．ｂは月曜日出席不可およびｃは火曜日出席不可であることがこのとき既にわかっている．また，このミーティングの場所の候補は，会議室２１２，会議室２１３である．会議室は空いているという条件がある．普通は，会議室は空いていると考えてよい．しかし，２１２会議室が月曜日に予約済みであること及び２１３会議室が火曜日に予約済みであることがこのとき既にわかっているものとする．出席者と開催場所と曜日を決定せよ．

Ｆｉｇ．２はこの例題を解くための知識ベースの例であり，対応するＲｅｔｅ－ｌｉｋｅネットワークをＦｉｇ．３に示す．Ｆｉｇ．３において，複数のブロックは環境を示し，ブロックの上側にはその環境のａｓｓｕｍｐｔｉｏｎｓスロットに格納されているビットベクタを十進表示し，ブロックの下側にはその環境のｎｏｄｅｓスロットから指示されるノードのデータを表示している．

Ｆｉｇ．３より，２入力ノード３８４がありかつその２入力ノードのｓｕｐｅｒリンクで指示される２入力ノード３８８に１入力ノード４から複数のトークンが流れ込むので推論実行時のラベル計算が効率的に行われる．

７・２　知識の追加

例えば，海側ラウンジがあり，ラウンジでもそのミーティングが可能であることがわかったとする．

Ｆｉｇ．４は追加知識ベースの例であり，これらを追加した後のＲｅｔｅ－ｌｉｋｅネットワークをＦｉｇ．５に示す．Ｆｉｇ．５より，節ｍｅｅｔｉｎｇと節ａｔ＿ｌｏｕｎｇｅとで，２入力ノード３８４が共有されるので追加知識に対しても，推論実行時のラベル計算が効率的に行われる．

７・３　知識の削除

次に，知識を削除した場合について示す．

例えば，ラウンジでそのミーティングを行った結果良

くなかったので，次回からのために知識ベースを変更しておきたいとする．そこで，この知識ベースからアトムｌｏｕｎｇｅ（Ｓ）を節の前件またはデフォルトの前提に含む知識を削除しておく方が推論効率の面から好ましい．本方法のインクリメンタル・コンパイラによれば，［ｌｏｕｎｇｅ（Ｓ）］を指定することにより，節ａｔ＿ｌｏｕｎｇｅを削除することができる（この例では一つの知識だけであるが，他にアトムｌｏｕｎｇｅ（Ｓ）を節の前件またはデフォルトの前提に含む知識が存在すればそのような知識もまた同時に削除される）．このときのネットワーク構造は，Ｆｉｇ．３に示す元のネットワーク構造に戻る．したがって，不要になったＲｅｔｅノードやリンクが残らない．しかも，このアトムを削除した効果は残り，このアトムを節の前件またはデフォルトの前提に含む知識を追加しても以後ネットワークには反映されない．すなわち，追加した知識だけでなく，削除した知識も，いわば学習効果としてシステムに残る．例えば，Ｆｉｇ．４と同様知識などは，誤ってもネットワークには反映されない．これは，ラウンジではそのミーティングを開催してはいけないという知識が追加されているとも考えられる．

７・４　コンパイル時間の評価

　上記例題について，ＰＳＩ－ＩＩ上のＡＰＲＩＣＯＴ／０を用いて，インクリメンタル・コンパイル機能を利用しない場合と利用した場合とのコンパイル時間について実験を行う．ここで，インクリメンタル・コンパイル機能を利用しない場合，入力知識ベースを修正して最初からコンパイルし直す方法をとった．

　実験の結果，７・２の例題をコンパイルする時間について，インクリメンタル・コンパイル機能を利用しない場合は３．０５［ｓ］，利用した場合は０．６９［ｓ］であった．推論時間は，両者とも０．５９［ｓ］であった．また，７・３の例題をコンパイルする時間について，インクリメンタル・コンパイル機能を利用しない場合は２．４９［ｓ］，利用した場合は０．０２［ｓ］であった．推論時間は，両者とも０．５３［ｓ］で変わらない．

　以上より，本インクリメンタル・コンパイラは，知識システムの段階的構築がコンパイル時間に関して極めて効率良く行え，更に推論実行効率を落とさないと結論付

けられる．この実験は小規模であるが，入力知識ベースがより大規模になれば本方法の効果は更に大きい．

8．むすび

以上，ATMSがRete-like ネットワークのインクリメンタル・コンパイラとして利用できることを示した．この方法によるインクリメンタル・コンパイラには，次の効果がある．

知識システムの段階的構築がコンパイル時間に関して効率良く行える．また，段階的構築のために，推論実行効率を落とさない．これは，知識の追加において共通のReteノードが共有されること，および知識の削除においても不要になったReteノードやリンクが残らないことによる．また，削除した知識は，有効にシステム内に保存され，入力知識ベースのメンテナンスを行う．この点に関して，本方法は，入力知識ベースのコンパイルとメンテナンスを同時に行う手法である．

また，仮説推論システムAPRICOT／0上に論理回路合成問題[12]に対する知識ベースをインプリメントし，特に回路の制約条件に関する知識の変更にこのインクリメンタル・コンパイラが有効であることを確認している．更に，コンパイルにATMSを用いることに対して，直接ESP言語にコンパイルする方式[3]と比較して，与えられた知識ベース全体をコンパイルする場合でもさほど時間がかかるものではないことも確認している．

尚，本方法は，Reteノードに環境すなわちビットベクタを割り当てているので，比較的容易に知識の共有関係をチェックできると思われる．この点に着目した，ネットワークの最適化等の研究は，今後の課題である．

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，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．

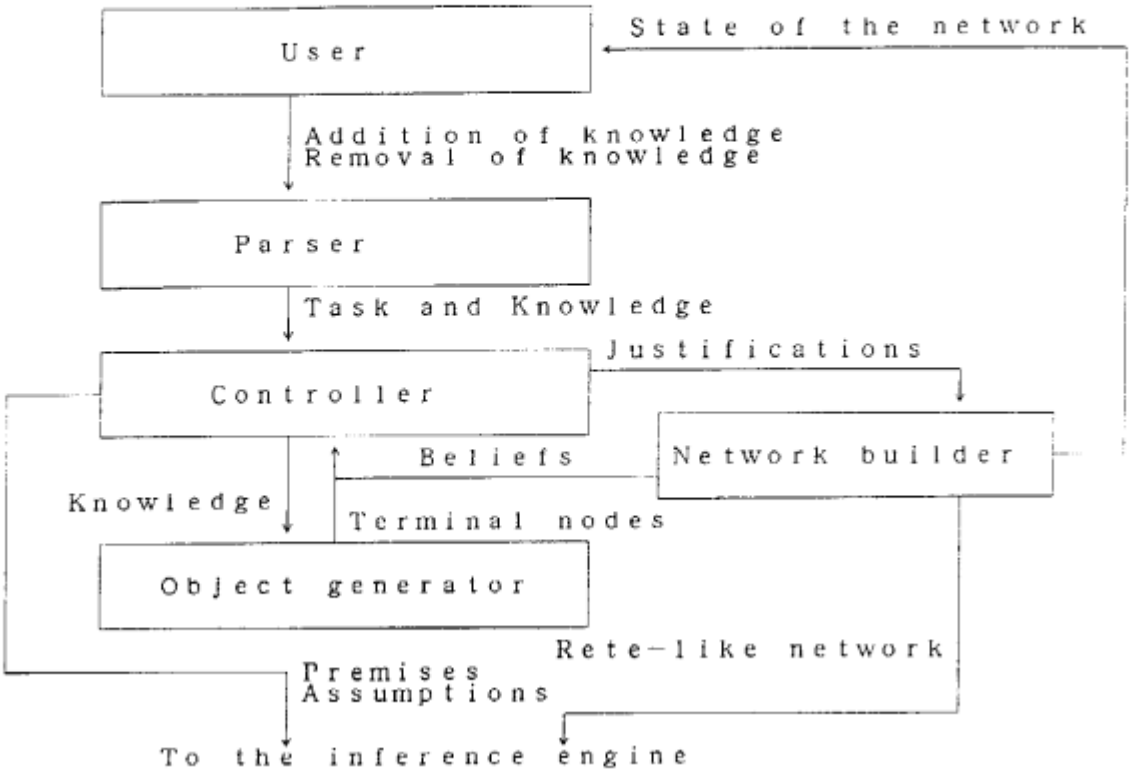
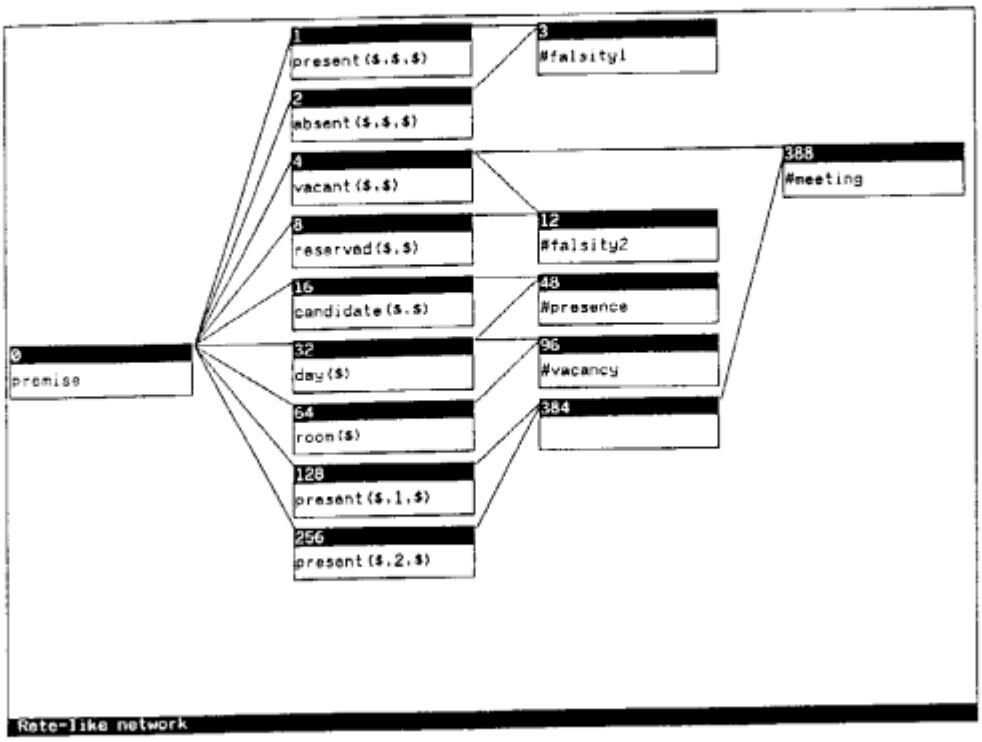


Fig. 1 Configuration of the incremental compiler.

```
falsity1::
        present (X, P, W),
        absent (X, P, W)
->
        [].

falsity2::
        vacant (N, W),
        reserved (N, W)
->
        [].

presence::
        candidate (X, P),
        day (W)
->
        assume (present (X, P, W)).

vacancy::
        room (N),
        day (W)
->
        assume (vacant (N, W)).

meeting::
        present (X1, 1, W),
        present (X2, 2, W),
        vacant (N, W)
->
        meeting (X1, X2, N, W).

candidate (a, 1).
candidate (b, 1).
candidate (c, 2).
candidate (d, 2).
room (212).
room (213).
day (monday).
day (tuesday).
absent (b, 1, monday).
absent (c, 2, tuesday).
reserved (212, monday).
reserved (213, tuesday).
```

Fig. 2 Initial knowledge base of the meeting plan.



The blocks show environments.

The upper part of the block shows a bit-vector expressed decimally.

The lower part of the block shows data of the modified ATMS nodes connected by "nodes" links with the environments.

———: The super (sub) link.

Fig. 3 Rete-like network of the initial knowledge base.

```
at_lounge::
      present(X1, 1, W),
      present(X2, 2, W),
      lounge(S)
->
      meeting(X1, X2, S, W).

lounge(seaside).
```

Fig. 4 Additional knowledge base of the meeting plan.

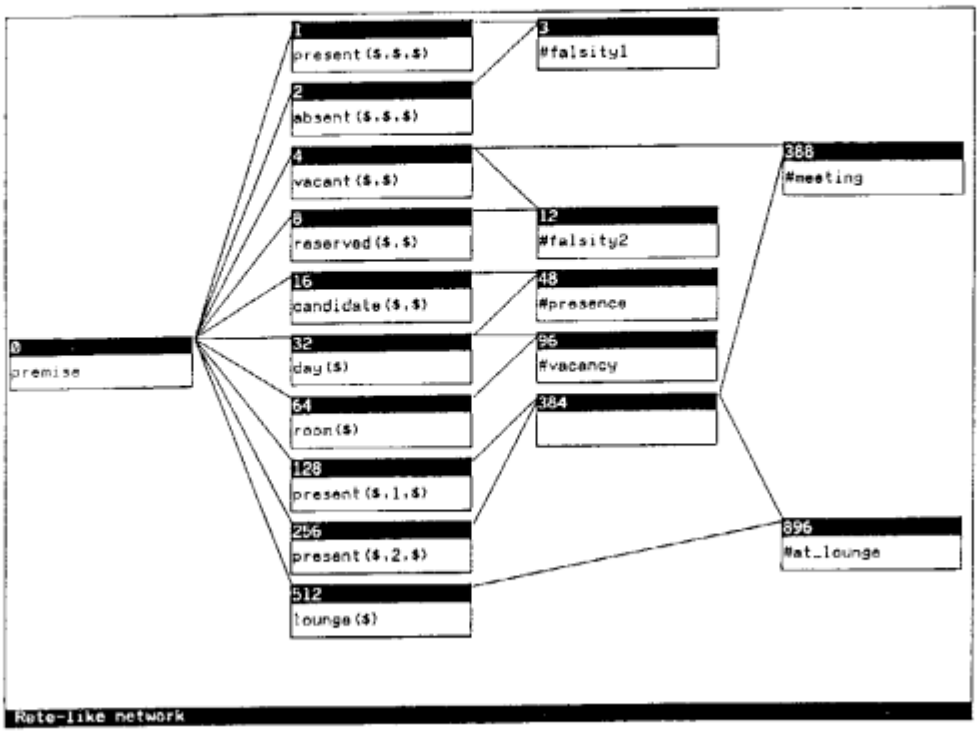


Fig. 5 Rete-like network of the initial knowledge base with the additional knowledge base.

◇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）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，人工知能学会誌投稿中，Ｎｏ．０３４（１９９０）．

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

（５）Ｒｅｉｔｅｒ，Ｒ．：Ａ Ｌｏｇｉｃ ｆｏｒ Ｄｅｆａｕｌｔ Ｒｅａｓｏｎｉｎｇ，Ａｒｔｉｆ．Ｉｎｔｅｌｌ．，Ｖｏｌ．１３，ｐｐ．８１－１３２（１９８０）．

（６）Ｆｏｒｇｙ，Ｃ．Ｌ．：Ｒｅｔｅ：Ａ Ｆａｓｔ

Algorithm for the Many Pattern／Many Object Pattern Match Problem, Artif. Intell., Vol. 19, pp. 17－37（1982）.
（7）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）.
（8）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）.
（9）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）.
（10）Chikayama, T.：Unique Features of ESP, Proc. of the International Conference on Fifth Generation Computer

Ｓｙｓｔｅｍｓ，ｐｐ．２９２－２９８（１９８４）．
（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）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）．