論　文

Technical Papers

推論過程からの概念学習(1)
—類型的推論過程の抽出—

CLIP : Concept Learning from Inference Pattern (1)
—A Method to Find Typical Inference Pattern—

吉田　健一*　元田　浩*
Ken'ichi Yoshida　Hiroshi Motoda

* 日立製作所基礎研究所
Advanaced Research Laboratory, Hitachi, Ltd., Hatoyama, Saitama 350-03, Japan.

1991年7月1日　受理

Keywords: hierarchical understanding, abstraction, knowledge representation, machine learning.

Summary

A new concept learning method CLIP (Concept Learning from Inference Pattern) is proposed. CLIP learns new concepts from inference patterns in contrast with the fact that most of the conventional concept learning methods learn a new concept from positive/negative examples. The learned concepts enables efficient inference on a more abstract level.

The learning process consists of the following two steps : 1) Convert the original inference patterns to a colored digraph, and 2) Extract a set of typical patterns which appears frequently in the digraph.

The basic idea is that : 1) The smaller the size of the digraph becomes, the less becomes the number of the data to handle and accordingly the more efficient becomes the inference that uses these data, 2) The reduced graph does not lose information, and the original information can be restored whenever needed, and 3) The reduced node represents a new cencept component (a new vocabulary).

A parallel-search algorithm based on "Pattern Modification" (mu-tation : to find a typical pattern), "Pattern Combination" (crossover : to mix patterns), and "View Selection" (to select a good set of patterns) extracts a set of typical patterns (chunks) which appear frequently in the digraph. The algorithm is similar to a previously reported Genetic Algorithm. In Pattern Modification, a partial digraph representing some meaningful component are extracted as a Pattern. In Pattern Combination, a new View is created as a set of Patterns. In View Selection, Views which result in smaller digraphs after being rewritten by the Patterns in the View are selected. Without this algorithm, we may have to rely on an exhaustive search which is computationally very expensive.

1. まえがき

人間は論理学や数学といった抽象的な概念を使ってさまざまな知見を得ることができる．このような抽象的概念抜きに現在の科学技術や文明を語ることはできないであろう．論理学や数学といった高尚な話を持ち出すまでもなく日常生活においても「あいつは頭が硬いから話にならないだろう」といった判断は常日頃行っている．この場合も「頭が硬い」は何らかの抽象的な概念を表しており，それを使うことによって余計な細部まで考えることなしに結論（例えば「話にならないから，相談するのはやめよう」）を得ることができる．

人間はどのようにしてこのような「概念」を獲得してきたのであろうか？

この疑問に答え，計算機の上にその仕組みを実現することは人工知能研究の重要な課題の一つである．

別の面から「概念」を考えてみよう．Fig. 1はCPUの一部である桁上げ演算回路を示している[1]．Fig. 2はその動作（電圧-電流の変化）を定性推論[2]によって

シミュレーションしたときの推論の様子を示している．Fig. 2 において，(a)，(b)，(c)はデータの整理・表示様式が異なるだけで推論状態（どのデータの値を使ってどのデータの値を推論したか）を示した図としては全く同じ図であることに注意されたい．(a)はデータの表示位置を無作為に決めた後，あるデータの値を別のデータから計算したことを示す矢印を示している．(b)はおのおののデータが定性推論の結果であることを利用して，縦軸にデータ名・横軸に時間をとって(a)を並べ換えたものである．より正確にいうと，横軸は定性推論でいうところの mythical causality[2]まで利用している．すなわち(b)は定性推論という領域固有の知識を利用して(a)を整理した図である．(c)はさらに桁上げ演算回路固有の知識まで利用して並べ換えた図である．図上端程回路の出力側のデータが並べてあり，下端程入力側のデータが並べてある．さらにNOR回路/NOT回路の固まりごとに整理もしてある．

どの図が一番見やすいであろうか？

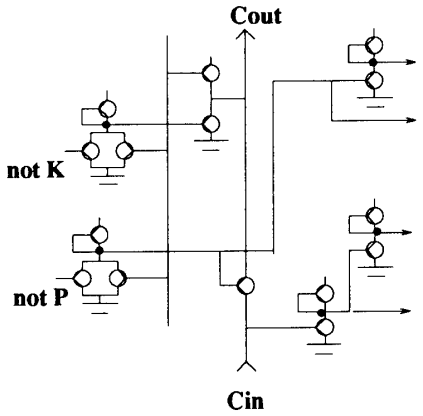
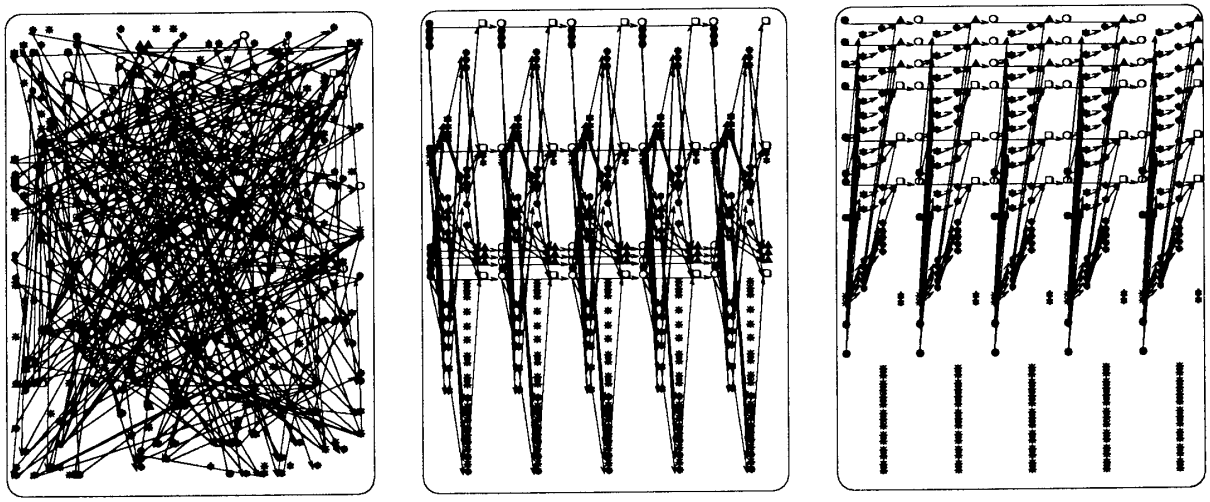


Fig. 1 Carry chain circuit.

著者らには圧倒的に(c)が見やすい．「どのデータからどのデータが決まっているか？」といった内容を説明したり，図を丸暗記する必要があるとすれば(c)を使うであろう．(a)を使うことはまずないと思われる．見やすい理由を内省してみると，ここでも「NOR の動作」・「NOT の動作」といった抽象的・論理的な概念を利用して図を見るときの注視点を調整しているためと思われる．また，図が実際に表している電圧・電流といった概念は使っていないように思える．すなわち，論理演算という，実際に表示されたものより抽象度が1段高い概念を使って図面の認識をしている．

以上の考察より著者らは，「概念」を「利用することにより推論が簡単になるもの」であると捉え，推論を簡単にするものを概念として学習する概念学習の方法を提案する．具体的には推論過程を分析し，類型的なパターンを抽出し，抽出したパターンを一つのマクロルール（学習した概念に相当）として学習する．この学習方法によれば，例えば電気回路を対象とした電圧・電流の変化に関する定性推論の過程を分析し，推論過程を簡単にする単語として，NOR/NOT に相当する単語を自動的に生成・学習できる．あらかじめ論理演算に相当する知識や概念形成のための特別な Heuristics を入力しておかなくても「推論を簡単にする」単語として類似の概念が生成される点が特徴である．さらに新しい単語を使った推論規則も生成するので，学習後は新しい概念を使った効率的な推論が可能となる点が帰納推論を中心とした従来の概念学習（例えば文献(3)）の研究と大きく異なっている．

本論文では，具体的な手法を中心に，簡単な実験結果も含めて報告する．また，別報[4]において，本手法に



(a) Naive inference pattern.

(b) Sorted by qualitative reasoning knowledge.

(c) Sorted by circuit knowledge.

Fig. 2 Example of concept usage.

より形成される概念構造に関して，実験結果を踏まえて考察を加える．

2. 色つき有向グラフ上での概念学習

本章ではCLIP(Concept Learning from Inference Pattern)の概要を説明する．CLIPは推論過程の分析結果から推論を効率的に行うという点で有効な概念を生成・学習する概念学習の方法である．学習過程は次の二つのステップから形成される．

1. 推論過程の色つき有向グラフ(Colored Digraph)への翻訳．
2. 色つき有向グラフの分析による類型パターンの抽出．

基本的なアイディアは，

- 推論過程を色つき有向グラフへ翻訳した場合，小さなグラフになる推論過程ほど推論の実行時に扱うべきデータ数が少なくなり，推論システムによる取扱いが簡単になる．
- 類型パターンを1点に縮約して小さなグラフに変換すれば，情報量を落とさず（縮約の逆操作で情報を復元可能）に，取扱いの簡単な推論過程を得ることができる．
- グラフ上で1点に縮約された類型パターンは新しい概念を表している．

ということである．Fig. 3はこのアイディアをまとめた図である．今，図左上に示した回路の挙動に関する推論が詳細に記述された知識を用いて行われたとする．図下端はこの過程を有向グラフで示したものである．具体的には，「左端の電荷 Q_1 が変化したという入力とコンデンサの挙動に関する推論規則2を用いて電圧 V_1 が，さらにトランジスタの挙動に関する推論規則3を用いて電流 I_1 の変化を予測した」といった推論過程が詳細な知識を用いた推論過程としてCLIPの入力となる．このグラフには「推論規則1の前に推論規則2,3,4が使われる」という類型的なパターンがある．CLIPはこのような類型的パターンを見つけることにより図右上端の図を出力する（正確には図を再現するのに必要な新しい推論規則も出力する）．ここでは新しい推論規則5が使われ電荷 Q_1 から電荷 Q_2 の値が直接求められている．右上端のグラフに対応する推論過程は下端のグラフに対応する推論過程より記憶容量などの計算コストが少なくてすむ．また推論規則5と推論規則1,2,3,4との対応関係を利用すれば，右上端のグラフから下端のグラフは復元できる．さらに下端のグラフが電圧・電流など物理量に関する推論であったのに対して，右上端のグラフは電荷の $[H/L]$ を真理値の［truue/false］と読み換えれば概念レベルが一つ上の抽象的な推論過程を表しているとみなせる．すなわち，CLIPにより見つけられた類型パターンはもとの世界には明示的には含まれていなかった新しい概念（この場合NOT演算）を表していることになる．

また，CLIPでは，類型パターンの抽出操作は色つき有向グラフの上で定義されており，色つき有向グラフへの翻訳が可能なものであれば，特定の専門領域に依存しない学習方法となっている．

2・1 色つき有向グラフへの翻訳

Fig. 4に推論過程（Inference Pattern）と，対応する色つき有向グラフ(Colored Digraph)，およびC言語でグラフを表現するのに用いた計算機上での実現方法（Digraph in Table）を示す．

CLIPでは，推論対象（例えば電気回路）はEntity-Relationship Model[5]で表され，推論は特定のデータ

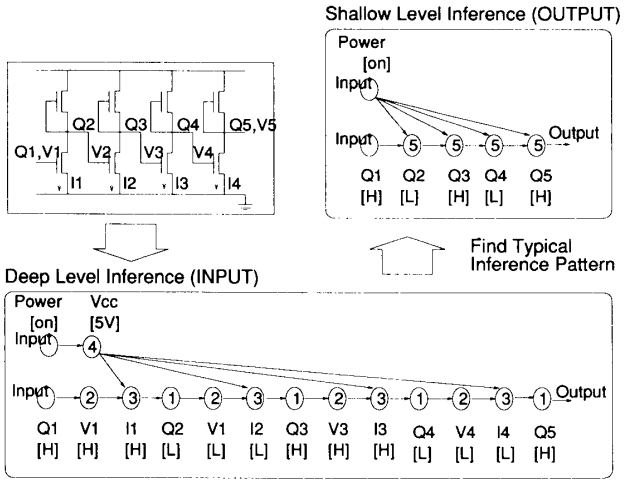


Fig. 3 CLIP : Concept Learning from Inference Pattern.

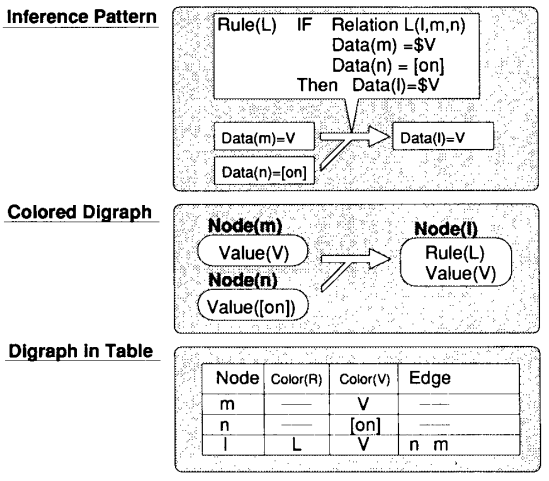


Fig. 4 Inference pattern, colored digraph and representation.

間の関係(Relationship)の利用方法を記述した推論規則(Interpretation Rule)によりデータ(Entity)の値が順次求められることで実行されるものと考える．電気回路の場合，回路の特性を記述した回路方程式が関係であり，個々の部位の電圧・電流の値がデータである．また，以下の説明では回路方程式を解釈して定性推論するための知識を表現したものを推論規則と呼んでいる．

Fig. 4 上段に示した推論過程に対応する色つき有向グラフがFig. 4 中段であり，Fig. 4 下段は対応する計算機上での実現方法である．データNにグラフの点(Node)Nが対応しており，点の色情報は，対応するデータの値と，値を求めるのに使ったRule番号である．Ruleに関する情報が辺(Edge)でなく点上に記憶されていることに注意されたい．また，辺は各点において何番目の辺であるかの情報も記憶されている．この順番は学習結果の利用の際に重要であるが詳しくは3•2節で説明する．

Fig. 5 に前述の桁上げ演算回路の動作を定性推論によってシミュレーションしたときの様子(一部のみ)を少し詳しく示す．左側のグラフはデータの依存関係と推論に利用した推論規則(図中の○□等の記号)を示しており，右側は対応するデータの値を示している．例えば図上端はopという部分の電圧v-opの初期値は[0](一つ目の値)で，推論規則□(左端の□：実際にはRule番号を用いている)と電流i-c-oppd(図2行目)i-e-oppu(図34行目)によって増加傾向(二つ目の値[s+])にあることが推論されており，次の時刻の値は[s+](三つ目の値)であることなどを示している．またこのグラフは，どのデータがどのデータとどの推論規則を用いて計算されたかという情報だけでなく，一つの推論規則の入出力関係の情報(どんな値を入力し，どんな値を出力するか．正確には推論過程で実際に利用されたサブセット)も含んでいる．例えば，推論規則□(定性的な足し算の規則)は図中で9回現れているが，図右側より対応する値の組合せは3種類(図中で使われている[vs−]+[s+]→[s+]/[0]+[s+]→[s+]/[−]+[s+]→[−])あることが読み取れる．すなわち図を分析することで，□という一つの推論規則が3種類の入出力関係(この場合定性的な足し算規則のサブセット)を持つことがわかる．

類型パターンの抽出操作には，このようなグラフを任意個入力する．すべてが同じ回路の動作のグラフであってもよいし，別の回路の動作のグラフが混じっていても構わない．複数の小さなグラフの集まりでもよいし，大きな一つのグラフでも構わない．同じグラフ

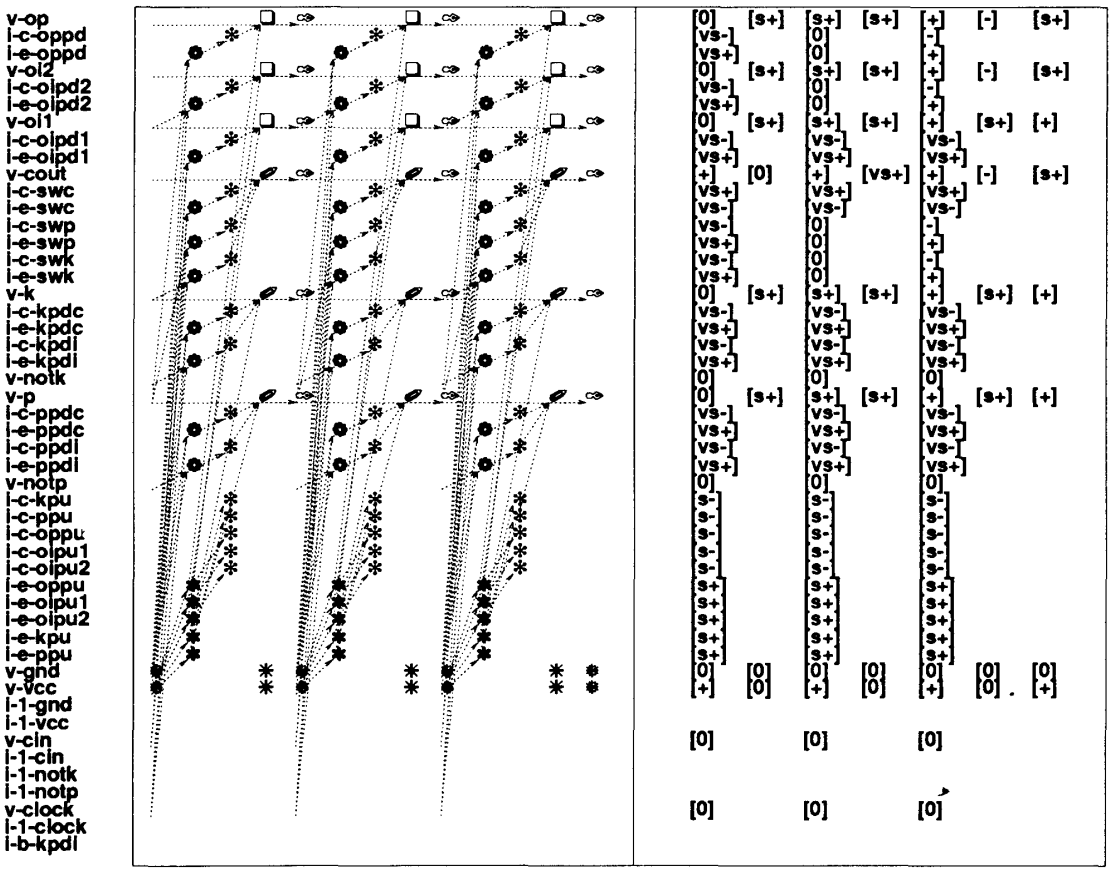


Fig. 5 Sample input digraph.

が多数含まれていてもよいし，一つだけでもよい（ただし，一つしか入力しなかったグラフに含まれる概念が重要でないと判断され捨てられる可能性はある）．ある時期推論システムが処理した結果をすべて引き渡すのでも構わない．通常の概念学習で必要な教師による教示は，推論システムが今後もよく処理するであろうケースを選び，学習時間を短くするためにあるので，すべて処理するつもりであれば必要ない．学習時間はグラフの大きさに比例する（3・3節参照）だけなので，人手を介してミスによる雑音が発生するよりは処理時間をかけたほうが好ましい場合もある．

この意味でCLIPは例題からの学習方法ではなく，学習・推論システムが置かれた環境からの学習方法であり，例題を作成する「教師」は不要である．また，類型パターンの抽出操作に渡されるのはFig. 2でいえば(a)に相当する情報だけであることに注意されたい．類型パターンの抽出操作はすべて色つき有向グラフ上の操作として定義されており，Fig 2(b)(c)に含まれている定性推論特有の情報や桁上げ連鎖回路特有の情報は使用されない．

2・2 類型パターンの抽出

色つき有向グラフが含む類型パターンの抽出操作はPattern Modification/Pattern Combination/View Selectionの三つの操作により実行される並列探索処理により実現されている．Fig. 6に概略手順を示す．

類型パターンの抽出操作では，類型パターンを記憶するためにViewとPatternという2種類のデータ構造を利用する．一つの類型パターンの情報を記憶するのがPatternであり，いくつかの類型パターンの組合せ方を記憶するのがViewである．回路の例では一つのPatternに特定の回路（例えばNORやNOT）の動作を推論したときの推論過程に対応するグラフのパターンが記憶される．Viewには複数のPatternを記憶し，処理対象のグラフを「NOR(の動作)の固まりの集まったもの」とみなすか「NOTの固まりとNORの固まりの混在したもの」とみなすか等，Patternの組合せ方を記憶する．

全体としては，まずPattern Modificationにより類型パターンの候補を作成し，Pattern Combinationにより組合せを調整するように動作する．

〔1〕 Pattern Modification

Pattern Modificationではまず，Viewを一つ選択し，その内容に従い処理対象の色つき有向グラフを書き換える．グラフの書換えでは，Viewに含まれるPatternごとにPatternと色情報まで含めて同じ構造を持つすべての部分グラフで，Patternと対応する点を1点に書き換える(Fig. 7(a)および(b)．図中で数字

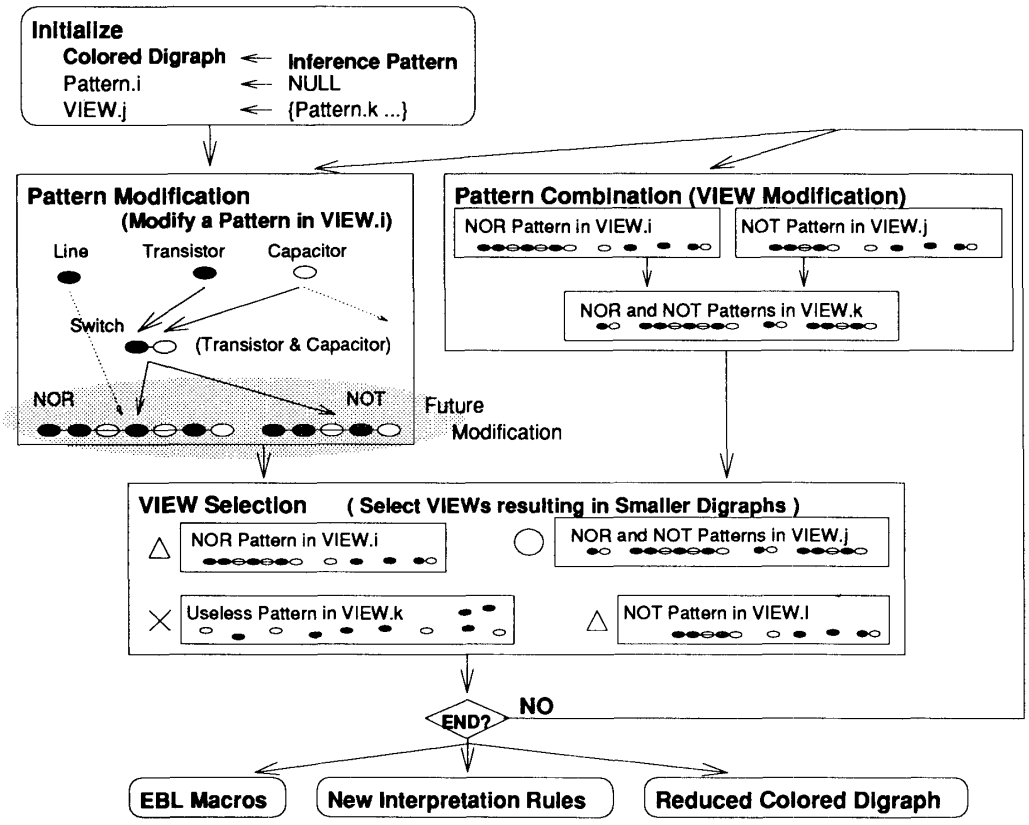


Fig. 6 Algorithm.

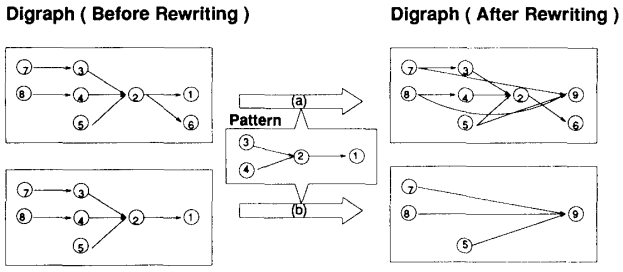


Fig. 7 Digraph rewriting by pattern substitution.

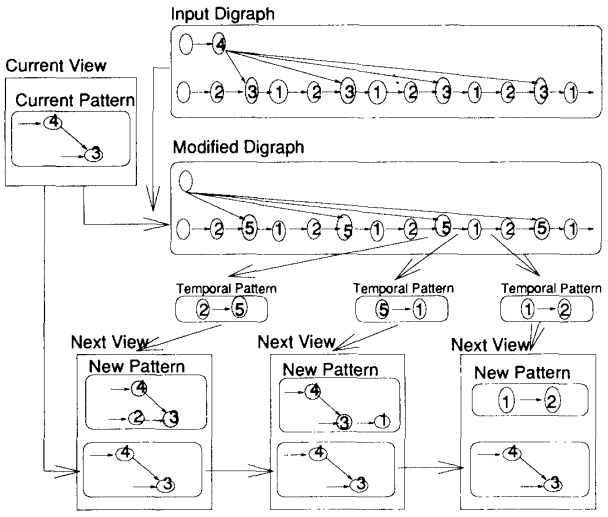


Fig. 8 Pattern modification.

は色番号)．ただし，途中の点が別のデータの計算に利用されている場合，もとの点を残す(Fig. 7(a))のでグラフ理論で言うところの縮約 (contraction) とは少し異なった操作である (Fig. 7(b)の操作だけなら縮約)．書き換えた点には新しい色を持たせる．次に書換え後のグラフを分析し「ある色の前に別の色がある」という小パターン (Temporal Pattern：1辺の両端の色のみに着目したパターン) を見つけ，見つけたすべての小パターンを新しい Pattern に翻訳して次の世代の View の候補として登録する (Fig. 8)．グラフにもとから含まれていた色からなる小パターンの場合，翻訳結果の Pattern は隣接する2点より構成されるパターンである．一方または両方が書換え後の新しい色を持つ場合，同じ小パターンでも翻訳結果は複雑な形状を表すことになる．ここで，Fig. 8 に示したように，修正前の Pattern も次の View の中に残すので正確には Modification ではない．これは修正後の Pattern が修正前の Pattern すべてをカバーせず，残したほうが評価値が良い場合があることによる．具体的には，大きな (中に含む点の数が多い，修正後の) Pattern によるグラフの書換え操作は，小さな (修正前の) Pattern による書換えより先に行う．修正後の Pattern が修正前の Pattern すべてをカバーしない場合，修正前の Pattern により再度書換え操作することで，修正後の Pattern だけで書換え操作するより小さなグラフを得ることができる．また，回路の例では色は回路方程式の番号であり*1，書換え後の点につける色は書換え前の複数の回路方程式から作られた新しい回路方程式につけられた番号ということになる．

View の記憶容量を越えて Pattern が作成された場合，あふれた Pattern はパターンの大きさ (中に含む点の数) が小さいものから順に捨てられる．また，Pattern Modification で一度に View に登録されるのは Pattern 一つのみである．Fig. 6 で"Future Modification"とあるのは，トランジスタやキャパシタの動作を記述した回路方程式の固まりに対応する Pattern から一度で NOR/NOT の固まりに対応する Pattern が作られるのではなく，後述する View Selection の操作を経て何度か Pattern Modification が繰り返される過程で作成されていくことを意味している．

〔2〕 Pattern Combination and View Selection

Pattern Combination では二つの View を選択し，各 View に含まれる Pattern の集合和を新しい View の要素とする．ここでも View の記憶容量を超えて Pattern が作成された場合，あふれた Pattern は小さなものから捨てられる．View の内容で書き換えられたグラフは大きさ (厳密には後述するようにほかの評価値も反映したもの．以下では特に断らない限り「グラフの大きさ」という場合 **3・1 節**で説明する方法で計算した値を意味する) が計算されている．View Selection では，Pattern Modification と Pattern Combination で作成された View のうち，書換え後のグラフの大きさが小さくなると予測される View を，初めに与えたパラメータ個選択する．予測値は Pattern Modification の場合，新しく見つけた小パターンの出現頻度を考慮してもとのグラフの大きさを修正した値

$$\left(1.0-\alpha*\frac{\text{グラフ中でのパターン A}\rightarrow\text{B の出現回数}}{\text{グラフ中の点 B の色の出現回数}}\right)$$

*(もとの View で書き換えて作ったグラフの大きさ)

を，Pattern Combination の場合，

小さいほうのグラフの大きさ$+\beta*$グラフの大きさの差

を用いた．α, β は実験ごとに初期パラメータとして与えた．

View Selection で予測値を用いるのはグラフの書換え操作が類型パターンの探索中，最も計算資源を要

*1 もう一つデータの値に対応する色も持っているが，ここでは利用しない．この色の利用方法は **3・1 節**で述べる．

求することによる．すなわち上記の処理では，予測した結果が良さそうなViewのみ実際に書換え操作をすることになる．なお，次の世代のViewを作成するときには初めに現在持っているViewすべてにおいてPattern Modificationを行い，次にやはり現在持っているすべてのViewの組合せに対してPattern Combinationを行う．処理順序をこうすることでPattern Combinationのときに利用するグラフの大きさは，Pattern Modificationの処理時に求めたものを利用できるようになり，必要なグラフの書換え回数を削減できる．

以上説明してきた類型パターンの抽出操作は，ViewをChromosome，PatternをGene，Pattern ModificationをMutation，Pattern CombinationをCrossover，グラフの大きさの評価式をFitness Function，View SelectionをSelectionと対応させれば，一種のGenetic Algorithm[6]とみなせる．すなわちFig. 9にデータの流れを示したように，全体としては，自然淘汰を利用した並列探索アルゴリズムとして動作している．

3. 実験結果

3・1 実験条件

以上説明してきた概念学習の方法をワークステーション上のC言語(約1 300行)で実現し実験した．結果をFig. 10に示す．

入力に用いた有向グラフはFig. 1に示した桁行げ連鎖回路の挙動に関する定性推論の結果であり，点の数2 176，辺の数2 144（点272個，辺268本のグラフが三つの入力の全組合せ8（$=2^3$）通り分で2 176＝8＊272，2 144＝8＊268)のグラフである．Fig. 10には

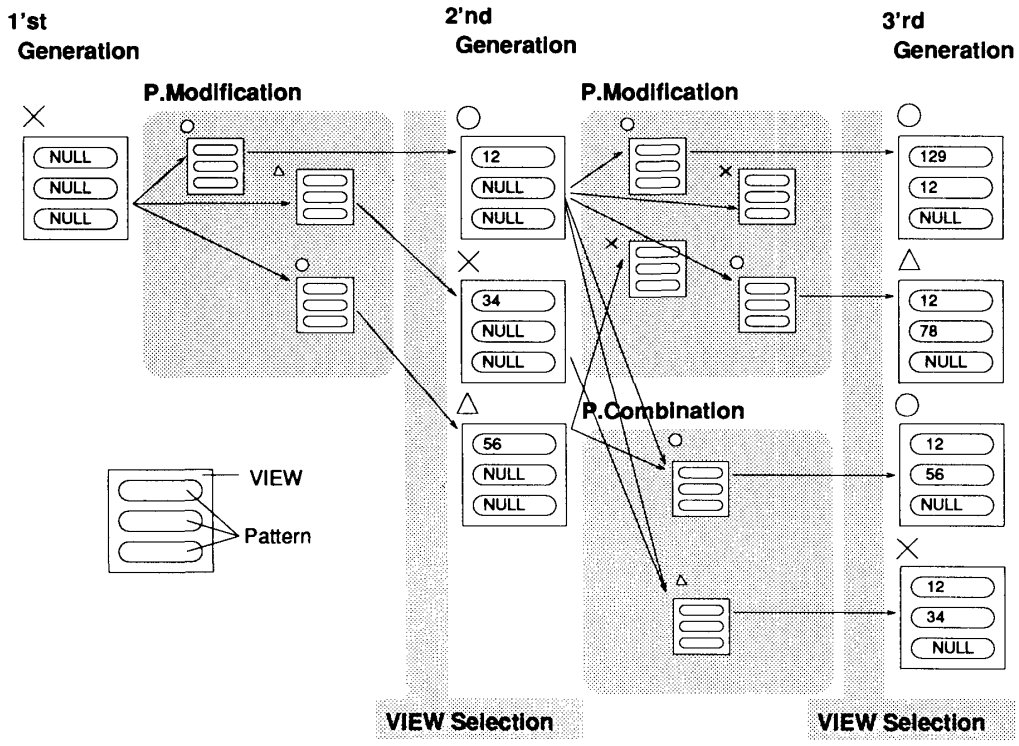


Fig. 9 Data flow.

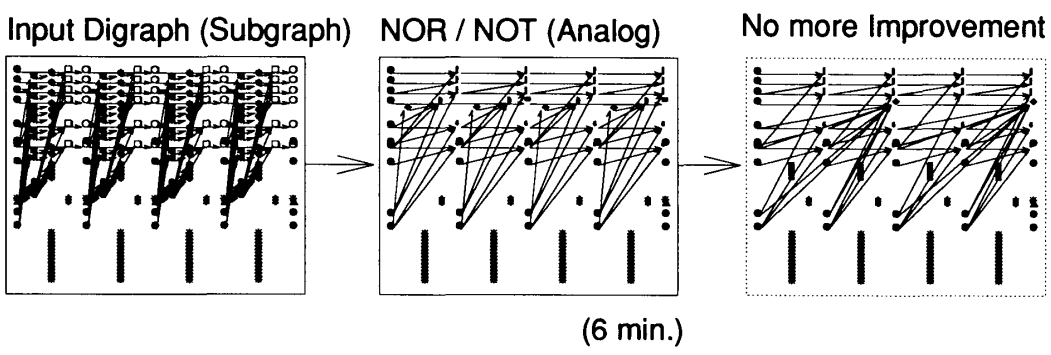


Fig. 10 Experimental result (resulting digraph).

全体の8分の1（1通り分）のみ左端のグラフとして示している．実際には同様の図を計8個入力した．定性値は[+],[0],[−]のほかに，大きさを考慮した[s+](small+),[vs+](very small+),[s−],[vs−]を扱い，「プルアップトランジスタに流れる電流はプルダウントランジスタに流れる電流より小さい」といった推論を行っている．グラフの大きさおよび予測値としては以下の3式を用いた．

グラフの大きさ＝
　点の数＋辺の数
　＋Σ推論規則の大きさ1
　＋Σ推論規則の大きさ2
　−Patternによるグラフの書換えが起こった回数　(1)

Pattern Modificationで使用する予測値＝

$$\left(1.0-0.1*\frac{\text{グラフ中でのパターンA}\rightarrow\text{Bの出現回数}}{\text{グラフ中の点Bの色の出現回数}}\right)$$

＊(もとのViewで書き換えて作ったグラフの大きさ)

Pattern Combinationで使用する予測値＝
　小さいほうのグラフの大きさ
　＋0.5＊グラフの大きさの差

式(1)において第1項「点の数」は推論システムの記憶容量への負荷，第2項「辺の数」は対象の記述と推論規則を用いて推論を行う場合に必要となるパターンマッチの負荷（すなわち最低限必要なUnificationの回数）に対応している．第3項「Σ推論規則の大きさ1」は各推論規則の入出力の対応の数のベキ乗の総和である．例えばNOTに関する推論規則は「False → True」「True → False」の2種類であるので4，NORは4種類なので16，推論規則がNOT/NORに関するものだけなら第3項は二つの和で20になる．この数が大きいほど推論システムは推論規則を記憶するための領域を必要とする．推論システムに計算資源を要求する点で，グラフの点や辺の数と同じなので，グラフの大きさの計算式に含めた．第4項「Σ推論規則の大きさ2」は各推論規則の入力数のベキ乗の総和で，NOTであれば$1^2=1$，NORであれば$2^2=4$，総和で5となる．この値も大きくなると推論時のパターンパッチに必要な処理量が増加する．第5項「Patternによるグラフの書換えが起こった回数」は，初めの世代では固まりを見つけても，十分意味を持つまで大きくないので，第3,4項が大きくなり，全体の評価値が悪くなるため導入した補正項であり，初めのうちは，多少ほかの評価値（特に第3,4項）が悪くても，何らかの書換えを行うものを次の探索の候補に残すように導入した．探索後半ではほかの評価値が悪いとこの項はほとんど0になるので（記憶されているPatternが大きく，ほかの評価値が悪いものでは書換えが起こらない），この補正項の影響は少ない．Pattern Modificationなど類型パターンの抽出操作では，原則として推論規則に対応する色だけを利用しているが，式(1)第3項「Σ推論規則の大きさ1」だけは値に対応する色を補助的な色として利用した．

3・2 学習結果

View内のPatternの最大数は7，同一世代内のViewの最大数は15を使用し，前述の類型パターンの抽出操作を50世代行った結果，25世代目にFig. 10中央のグラフが得られた（10 MIPSのワークステーションでCPU時間が約6分）．25世代目以降は式(1)の第3,4項が大きくなり，中央のグラフが最も小さなグラフと評価された．後のほうで得られたグラフをFig. 10右端に示す．点の数，辺の数が減り，書換えに利用した一つ一つのパターンが中央のものより大きくなっている（パターンに含まれる点の数が多い）のが図から読み取れるが，推論規則の大きさに対応する第3,4項は大きくなっている．

Fig. 10中央のグラフを得るのに利用したViewに含まれるPattern(NOT回路に対応）とPatternの情報からEBL[7]で得られるMacro Ruleおよび推論規則（Interpretation Rule）を機械的に作成したものをFig. 11に示す．Patternには六つの点が含まれており，順に①端子の電圧変化に関する関係式（による推論過程のノード），②端子の電圧変化と流入電流の関係式，③トランジスタのコレクタ-エミッタ電流の関係式，④エミッタ電流とベース電圧の関係，⑤電源電圧とエミッタ電流の関係，⑥電源に関する知識に対応している．③④はプルダウントランジスタ，⑤⑥はプルアップトランジスタに関する知識である．また，Macro Ruleの条件部の各関係式において右辺の何番目にデータがあるかと，Patternの中の各点で何番目の辺であるかとが対応している．推論規則は「入力がV_b=[+]，V=[0]であったとき，出力がその下のV_{next}=[0]であった」といった入出力の組合せを記憶している（図には一部のみ示した）．この回路の場合，電圧が[+]のとき論理値[true]，[0]のときが[false]といった使われ方をするので，図示したように[s+]などを含むのはNOTの推論規則としてはおかしい．これに関しては別報[4]で詳しく考察を加える．

なお，Fig. 11にはNOT回路の回路方程式による推論を1まとまりとして取り出したPattern，Macro Rule，推論規則のみ示したが，実際にはNOR回路に

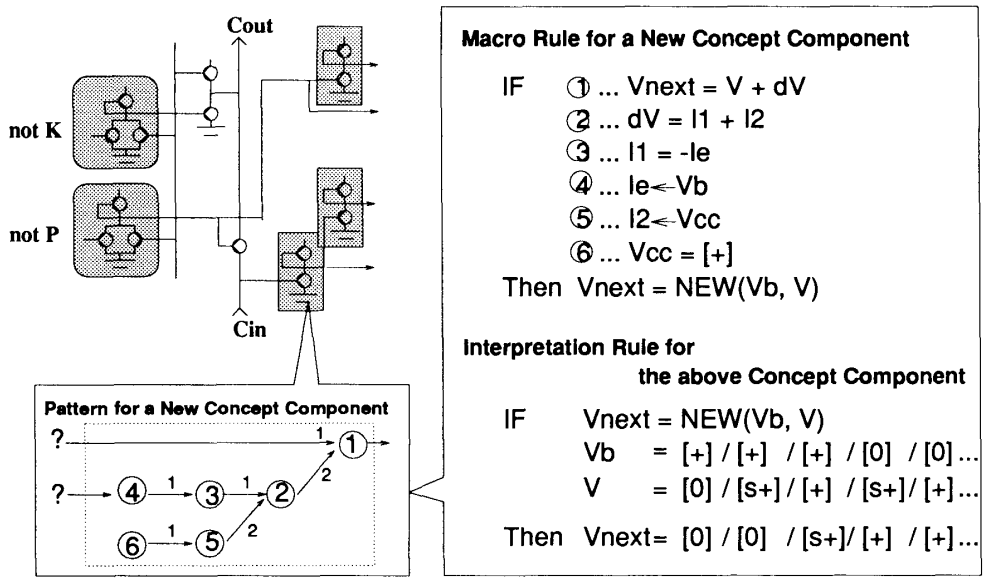


Fig. 11 Resulting rule.

対応する Pattern も取り出せている．

3・3 処理時間および必要な記憶容量に関する考察

CLIP は Unification の代わりに辺の対応関係を利用して高速化した EBL システムともみなせる*2．Fig. 12 に C 言語で Pattern を実現したときの配列の構造を示す．Fig. 12 に示したように Pattern は長さ $F(D, W)$ の大きさの配列で記憶可能である．ここで

$$F(W, D)=1+W*F(W, D-1) \quad \text{if} \quad D>1$$
$$=1 \quad \text{if} \quad D=1$$

D はパターンの深さ，W はパターンの幅であり Fig. 10 の探索では，W は 4，D は 6 を利用した．0 は任意の色 (EBL と対応させる場合は任意の Rule) を意味しており，グラフの各点で何番目の辺であるかを示す番号が EBL での変数名に相当している．EBL では変数を扱うために Unification(または論理的に等価な操作) を利用する必要があったが，類型パターンの抽出中の CLIP では数値の比較だけで等価な操作を行っている．すなわち Fig. 12 に示したパターンが Fig. 4 下端に示した配列の特定位置にあるか否かの判定は，深さ D のパターンが配列のあるインデックス位置にあるかどうかの再帰プログラムとして効率良く実現できる．

実験条件を変えたときに処理時間に最も影響を与えたのは入力グラフの大きさで，両者はおおよそ比例関係にあった．当初「Pattern が大きくなるほどマッチング処理に時間がかかり，全体の処理時間が増加する」

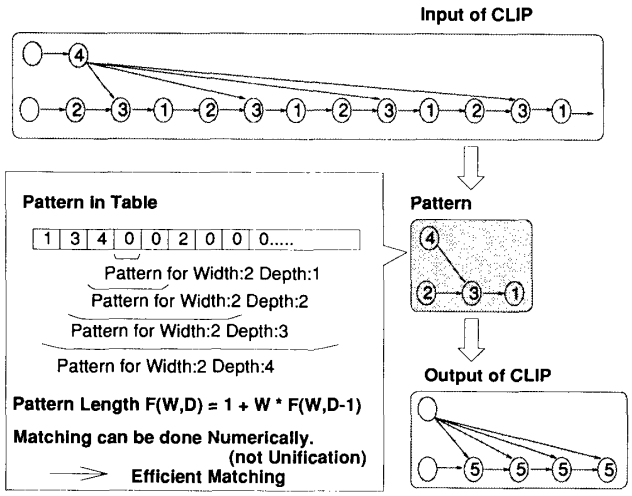


Fig. 12 Pattern representation.

といった予想を立てていたが，予測したほど影響はなかった．Pattern や View に含まれる点の数*3の世代ごとの平均値と，一つの View で入力グラフの書換え操作をするのに必要な平均 CPU 時間を計測した結果を Fig. 13 に示す．世代が進むにつれて Pattern や View に含まれる点の数は増えているが，CPU は増えていないことが読み取れる (Fig. 13 で第 1 世代の CPU 時間が極端に短いのは第 1 世代の View は何も Pattern を含まず，書換えが実行されないことによる)．これは小さなときはマッチする確率が高く書換え処理に時間を消費するため両方の処理時間が打ち消しあったものと思われる．実験条件を変えた計測でも，この傾向は変わらず，処理時間に最も影響を与えたのは入力グラフの大きさであった．

なお，Fig. 12 に示した Pattern の実装方法は，パターン幅と深さを少し大きくするだけで必要な記憶容量

＊2 EBL と Unification の関係は文献(8)に詳しい．CLIP と EBL の比較は**次章**で述べる．

＊3 View は複数の Pattern から構成されているので，間接的に点を含んでいることになる．

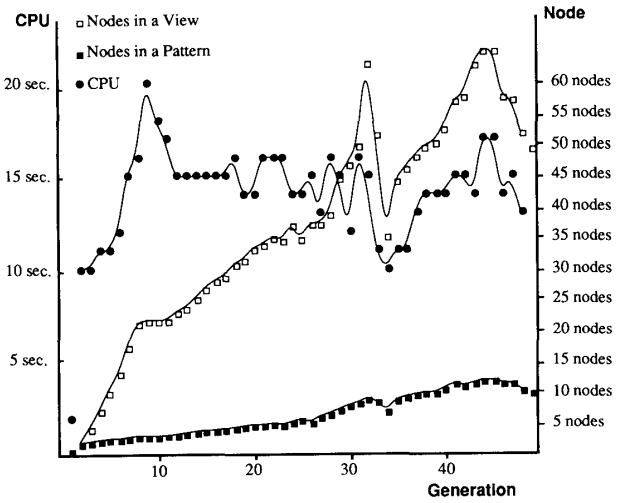


Fig. 13 Relation between pattern length and CPU time.

が著しく増加する（本報で説明した実験ではプロセスサイズが5 MByte）．しかし，内容のほとんどは0の配列であるので0以外の値とそのインデックスの組合せで記憶するなど削減は簡単である．値とそのインデックスの組合せで記憶した場合，本報で説明した実験に必要な記憶容量はプロセスサイズが1 MByte以下になる．またPatternに含まれる点の数が100を超えるような概念が必要となる場合は考えにくいが，仮に100とおいても，現在より小記憶容量の実現が可能である．

4. 関連研究との比較

推論過程を分析し概念を学習しようとする試みは，EBLの研究[7]の中でも扱われている．両者はチャンキングメカニズムを利用して概念を学習しようという点では同じであるが，次の2点で大きく異なっている．

- EBLとCLIPの一番の違いは，CLIPが自動的に固まりを切り出す能力を持っていることである．CLIPは，グラフの色として推論規則を用いた場合，Goal ConceptとOperational Criteriaを自動的に調整しUtility問題[9]に対処するEBLシステムとみなせる．EBLでは両者は入力であるが，CLIPでは出力である（Table 1）．
  概念学習システムとして見た場合，EBLのように両者が入力となっているシステムでは概念の固まりを見つけているのは利用者であり，システムは単に一般化した形で記憶しているのに過ぎない．一方，CLIPでは，システム側でどこまでを固まりとみなすかを提案している（**3・3節**で述べたように，一般化の仕組みは同等である）．

Table 1 The differences between EBL and CLIP.

	EBL	CLIP
Input	• Goal Concept • Training Example • Domain Theory • Operationality Criterion	• **Plural** Training Examples • Domain Theory • **Characteristics of the Inference Engine**
Output	• A Macro Rule	• **A Set of** Macro Rules **(Operationality Criterion** Implicitly Embedded) • **Interpretation Rule :** (New Domain Theory for the Abstract Level Inference) • **Goal Concept**

例えば，文献(7)では，“cup”という概念をEBLで学習させた例が説明されているが，“cup”という概念を学習するためには，“liftable”という概念が重要なのか，“light”や“handle”という概念が重要なのかはOperational Criteriaとして利用者が入力している．また，“cup”という概念を学習すべきだとの指摘も利用者が行っている．複雑な概念を学習するときには，CLIPのような両者をデータから自動的に調整する仕組みが必要である．

- 大型計算機を構成する回路のような複雑な対象を人間が扱う場合，扱おうとする問題に応じて対象を記述するEntity-Relationship Model(**2・1節**)を切り換えていると思われる．例えばタイミングやファンアウトの問題を考察するときには回路方程式が各Entity間のRelationshipを与えるために利用され，論理的な挙動を問題にするときには論理方程式が用いられる．回路方程式や論理方程式を使って推論するためには数学や論理学の知識が用いられる．CLIPで学習されたMacro RuleはEntity-Relationship Modelを切り換えるために利用され，Interpretation Ruleは各Modelでの推論を行うために利用される．
  EBLのMacro Rule(Fiig. 11)を用いても，与えられた記述（CLIPではEntity-Relationship Modelで与えられる）をレベルの異なる概念に変換することができる．文献(7)の例では，“handle”や“light”から構成された記述を，“cup”という上位レベルの概念に変換できる．しかし，上記のような意味での概念レベルが変わった世界（“cup”で記述されなおした世界）で推論（例えば「“cup”を使って水が飲める」）を行うことまでは扱われていない．CLIPでは変換結果の概念を利

用して推論を行うための推論規則（Fig. 11 Interpretation Rule）まで出力するので，新しい概念を使った推論（「notの入力が[false]だったので…」等）を行うことができる．

5. むすび

推論過程の分析結果から推論を効率的に行うという点で有効な概念を生成・学習する概念学習の方法（CLIP：Concept Learning from Inference Pattern）を提案した．この学習方法によれば，例えば電気回路を対象とした電圧・電流の変化に関する定性推論の過程を分析し，推論過程を簡単にする単語として，NOR回路/NOT回路に相当する単語を自動的に生成・学習できる．あらかじめ抽象度の高い概念を形成するための特別のHeuristicsを入力しておかなくても「推論を簡単にする」単語として類似の概念が生成される点が特徴である．

ワークステーション上にC言語を用いて実現したプログラムによる実験結果では，CPUの一部である桁上げ演算回路の挙動の定性推論結果を分析することで，回路上のNOR/NOT部分を1まとまりとして切り出し，その部分の挙動に関する推論規則を生成できることがわかった．

◇参考文献◇

（1） Mead, C. and Conway, L.：*Introduction to VLSI Systems.*, Addison-Wesley Publishing Company (1980).

（2） De Kleer, J.：A Qualitative Physics Based on Confluences, *Artif. Intell.*, Vol. 24, pp. 7-83 (1984).

（3） Cohen, P. R. and Feigenbaum, E. A., ed.：*The handbook of artificial intelligence*, chapt. 12, William Kaufmann, Inc. (1982).

（4） 吉田，元田：推論過程からの概念学習（2）概念構造の構成要因，人工知能学会誌，Vol. 7, No. 4, pp. 686-696 (1992).

（5） Chen, P. P.：The Entity-Relationship Model Towards a Unified View of Data, *ACM Transaction of Database Systems*, Vol. 1, No. 1, pp. 9-36 (1976).

（6） Goldberg, D. E.：*Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company (1989).

（7） Mitchell, T., M. Keller, R. M., and Kedar-Cabelli, S. T. Explanation-Based Generalization：A Unifying View, *Machine Learning*, pp. 47-80 (1986).

（8） Mooney, R. J. and Bennett, S. W.：A Domain Independent Explanation-Based Generalizer, *AAAI-86*, pp. 551-555 (1986).

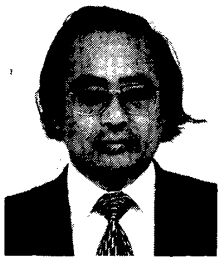
（9） Minton, S.：Quantitative Results Concerning the Utility of Explanation-Based Learning, *AAAI-88*, pp. 564-569 (1988).

〔担当編集委員・査読者：石塚　満〕

著者紹介

吉田　健一（正会員）

1980年東京工業大学理学部情報科学科卒業．同年，(株)日立製作所に入社．同社エネルギー研究所にてプラントの異常診断などの研究に従事．1986年より，基礎研究所にて知識表現，機械学習，定性推論などの研究に従事．1984年日本原子力学会論文賞，1990年電気学会論文賞，1991年人工知能学会全国大会優秀論文賞受賞．情報処理学会，AAAI，ACM各会員．



元田　浩（正会員）

1965年東京大学工学部原子力工学科卒業．1967年同大学院原子力工学専攻修士課程修了．同年，日立製作所に入社．同社中央研究所，原子力研究所，エネルギー研究所を経て，現在，基礎研究所主管研究員．原子力システムの設計，運用，制御に関する研究，診断型エキスパートシステムの研究を経て，現在は人工知能の基礎研究，特に機械学習，知識獲得，知識コンパイル，定性推論などの研究に従事．工学博士．元日本ソフトウェア科学会理事，現人工知能学会理事，Knowledge Acquisition (Academic Press) 編集委員，IEEE Expert編集委員，コンピュータ科学編集委員．1970年日本原子力学会奨励賞，1977，1984年日本原子力学会論文賞，1991年人工知能学会論文賞受賞．情報処理学会，日本ソフトウェア科学会，日本認知科学会，日本原子力学会，AAAI，IEEE Computer Society各会員．